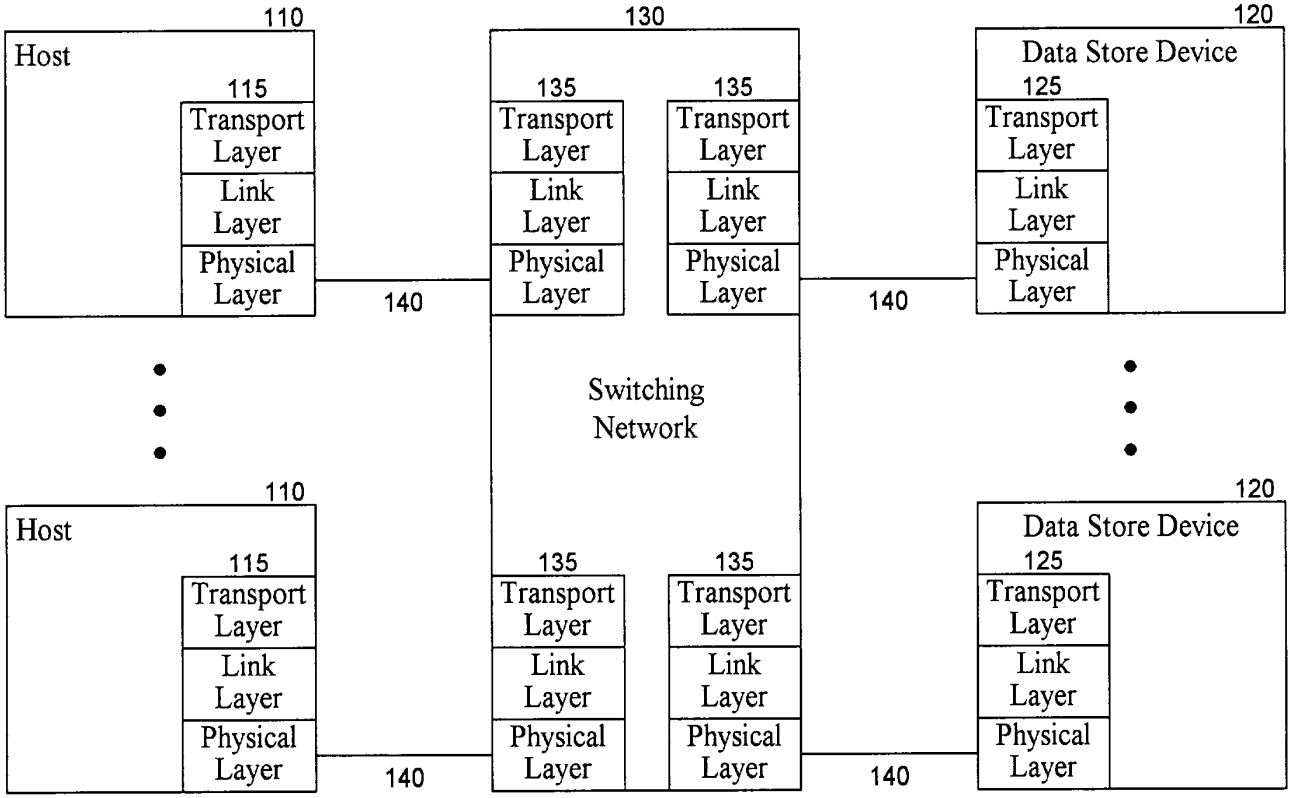

110
Host
115
Transport Layer
Link Layer
Physical Layer
140
130
135
Transport Layer
Link Layer
Physical Layer
135
Transport Layer
Link Layer
Physical Layer
Switching Network
140
120
Data Store Device
125
Transport Layer
Link Layer
Physical Layer
110
Host
115
Transport Layer
Link Layer
Physical Layer
140
135
Transport Layer
Link Layer
Physical Layer
135
Transport Layer
Link Layer
Physical Layer
140
120
Data Store Device
125
Transport Layer
Link Layer
Physical Layer

Fig. 1

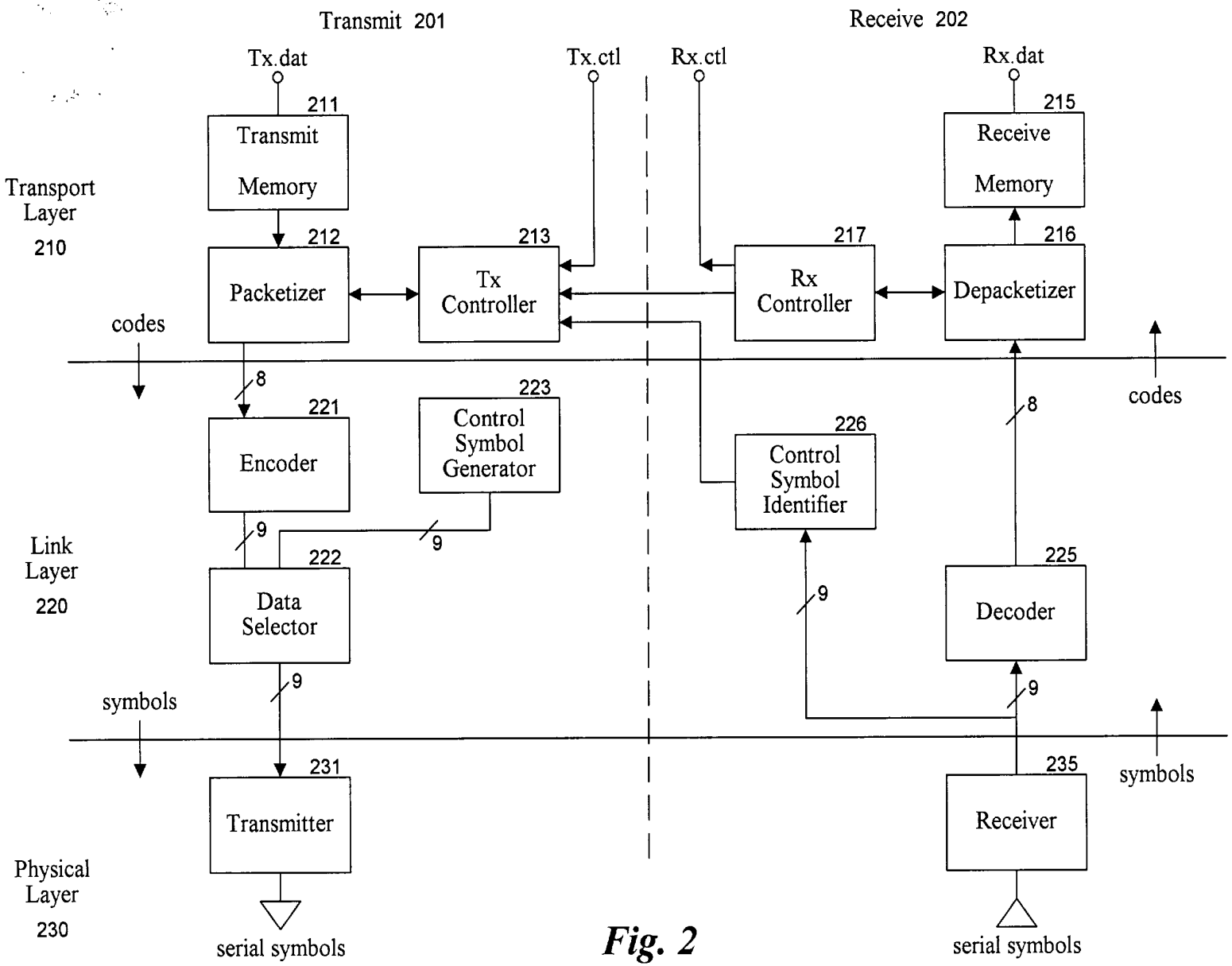

Transmit 201
Receive 202
Tx.dat
Tx.ctl
Rx.ctl
Rx.dat
211
Transmit Memory
215
Receive Memory
Transport Layer 210
212
Packetizer
213
Tx Controller
217
Rx Controller
216
Depacketizer
codes
codes
8
8
223
Control Symbol Generator
221
Encoder
226
Control Symbol Identifier
9
9
9
9
Link Layer 220
222
Data Selector
225
Decoder
symbols
symbols
231
Transmitter
235
Receiver
Physical Layer 230
serial symbols
serial symbols

Fig. 2

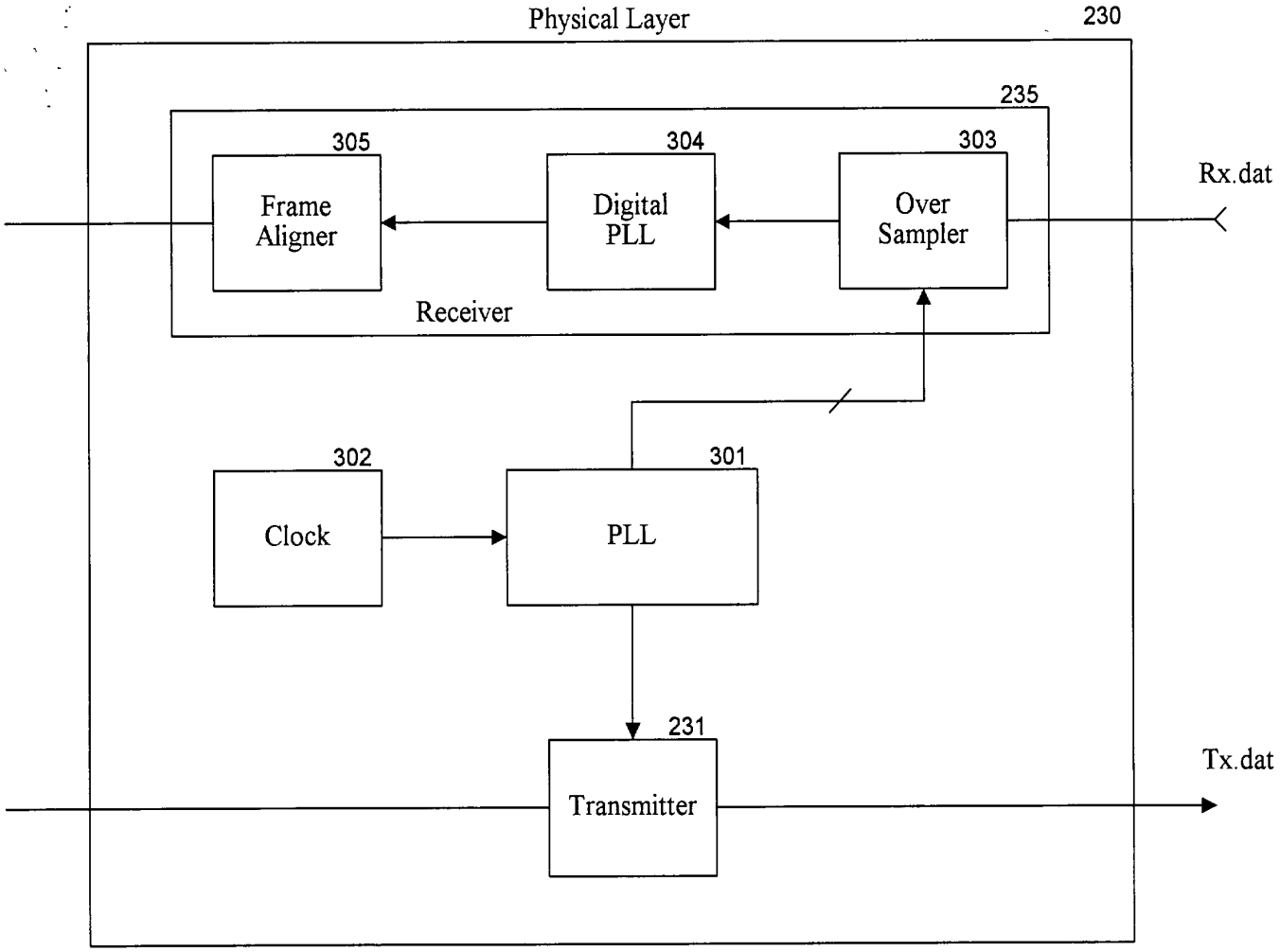

Physical Layer
230
235
305
304
303
Frame Aligner
Digital PLL
Over Sampler
Rx.dat
Receiver
302
301
Clock
PLL
231
Transmitter
Tx.dat

Fig. 3

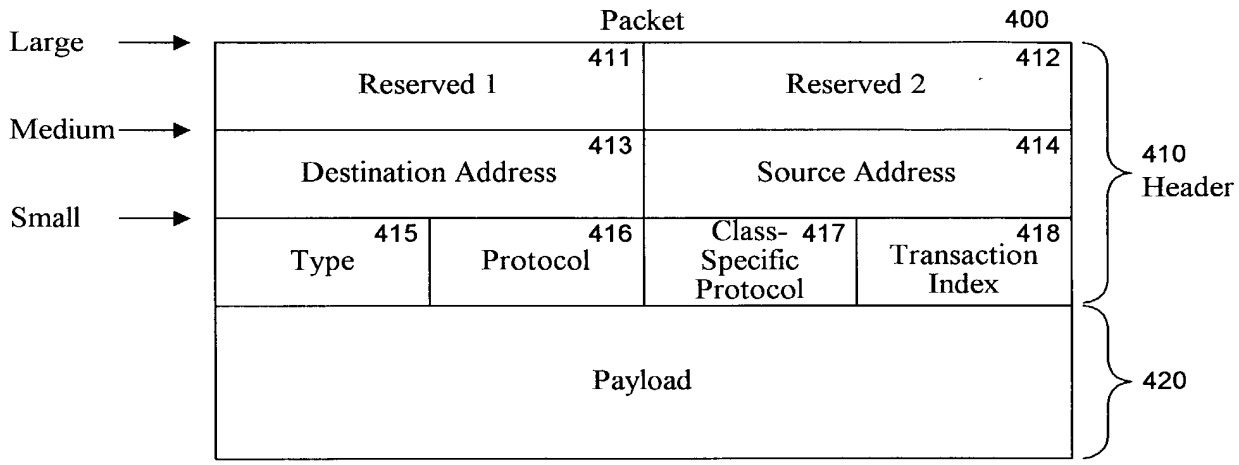

Packet
400
Large
Medium
Small
411
Reserved 1
412
Reserved 2
413
Destination Address
414
Source Address
415
Type
416
Protocol
Class- 417
Specific
Protocol
418
Transaction
Index
410
Header
Payload
420

Fig. 4

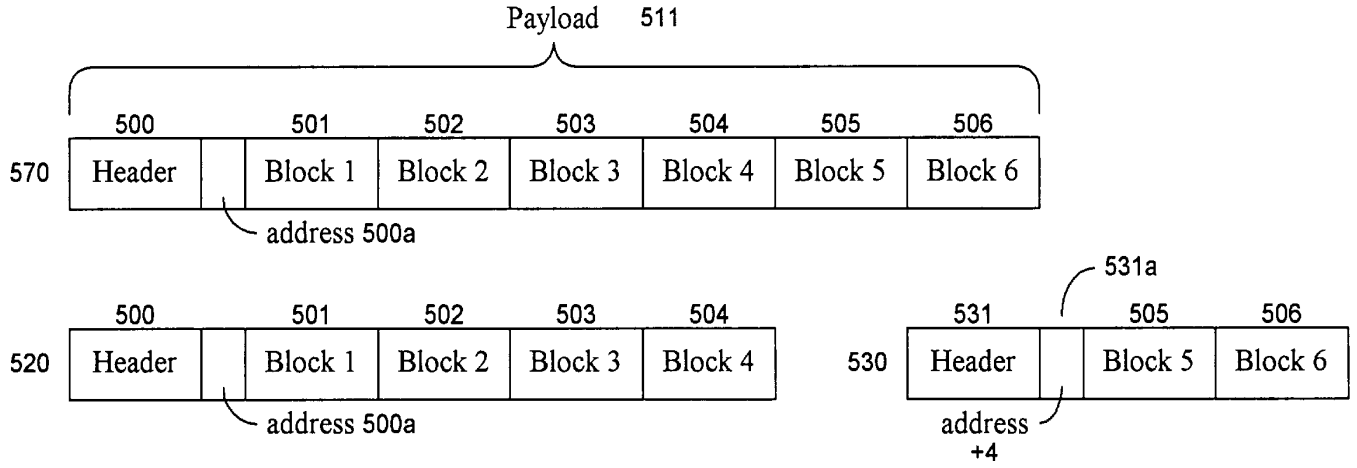

Payload 511
500
501
502
503
504
505
506
570
Header
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
address 500a
500
501
502
503
504
520
Header
Block 1
Block 2
Block 3
Block 4
address 500a
531a
531
505
506
530
Header
Block 5
Block 6
address
+4

Fig. 5

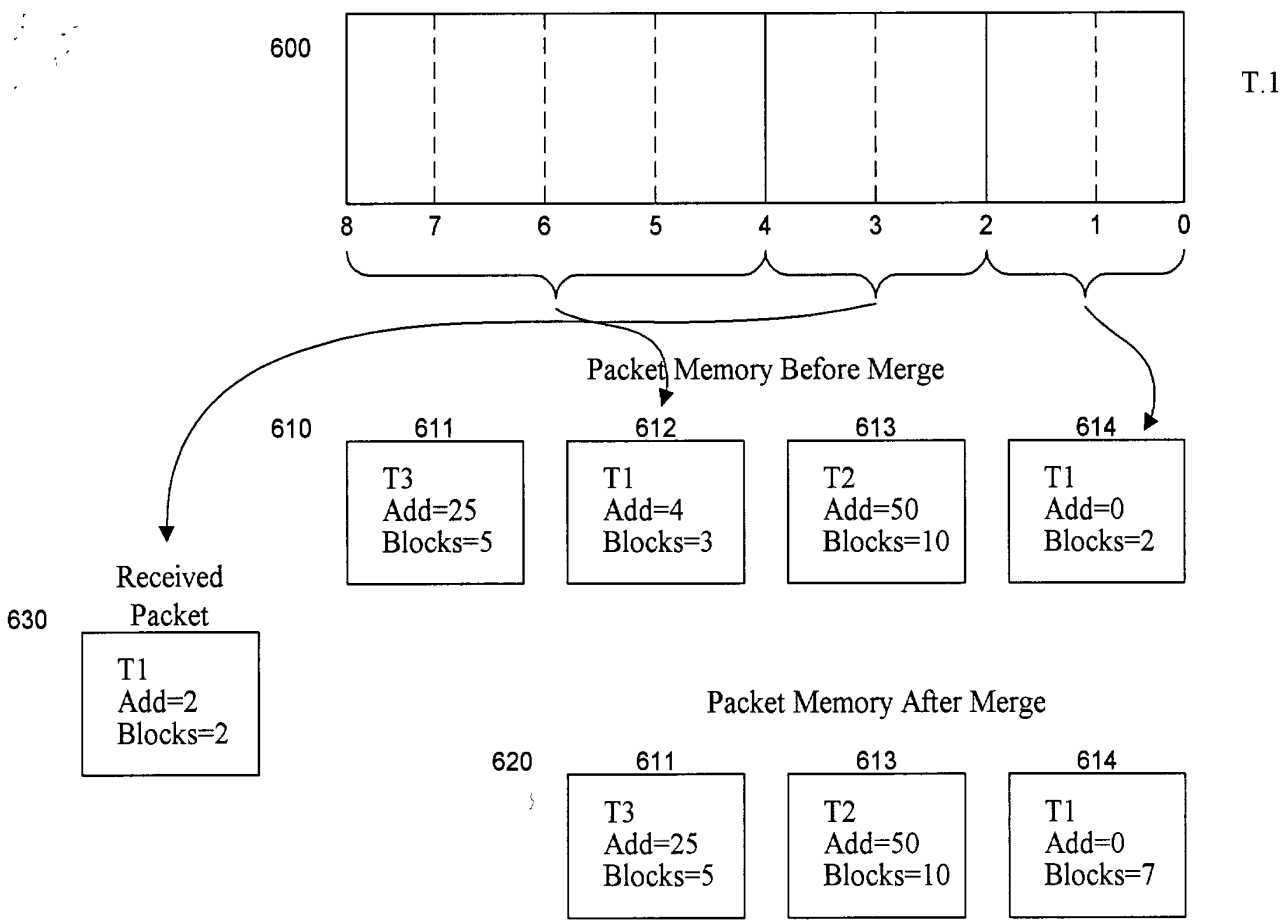

600
T.1
8
7
6
5
4
3
2
1
0
Packet Memory Before Merge
610
611
612
613
614
T3
Add=25
Blocks=5
T1
Add=4
Blocks=3
T2
Add=50
Blocks=10
T1
Add=0
Blocks=2
Received
Packet
630
T1
Add=2
Blocks=2
Packet Memory After Merge
620
611
613
614
T3
Add=25
Blocks=5
T2
Add=50
Blocks=10
T1
Add=0
Blocks=7

Fig. 6

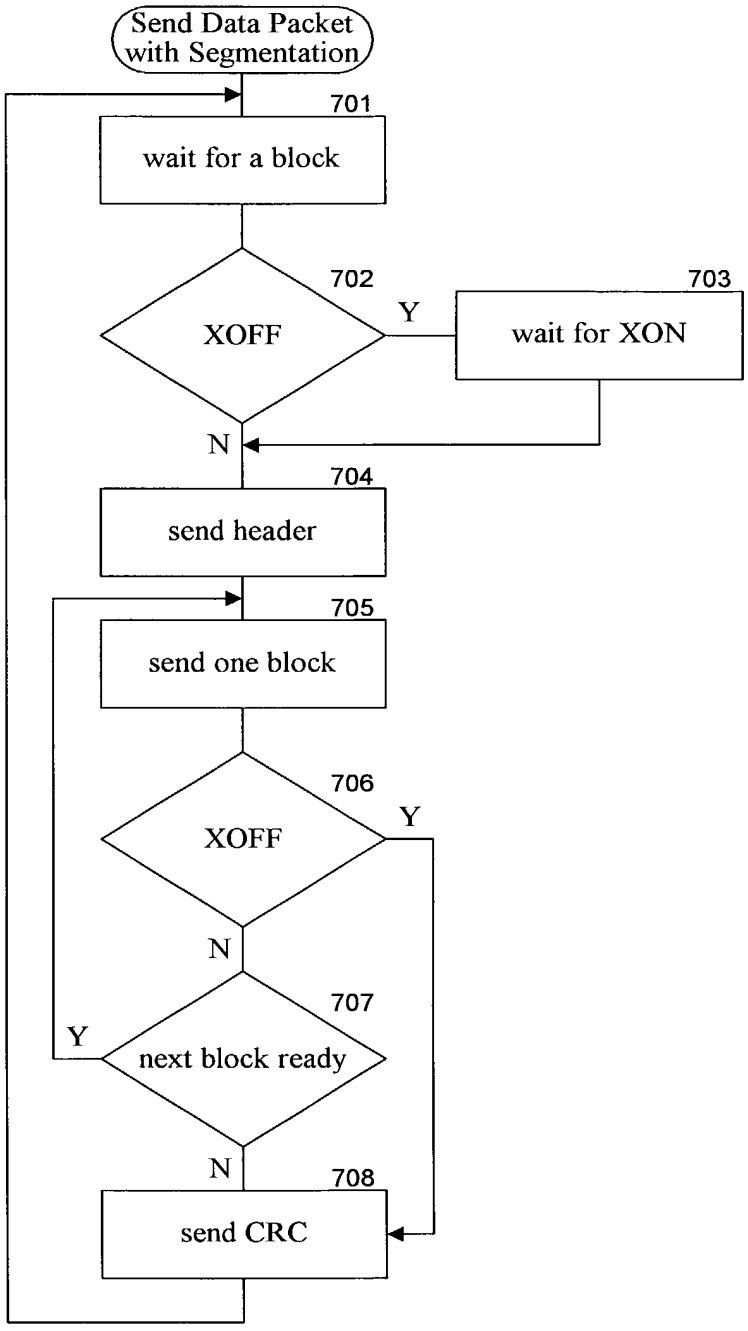

Send Data Packet with Segmentation
701
wait for a block
702
XOFF
Y
703
wait for XON
N
704
send header
705
send one block
706
XOFF
Y
N
707
Y
next block ready
N
708
send CRC

Fig. 7

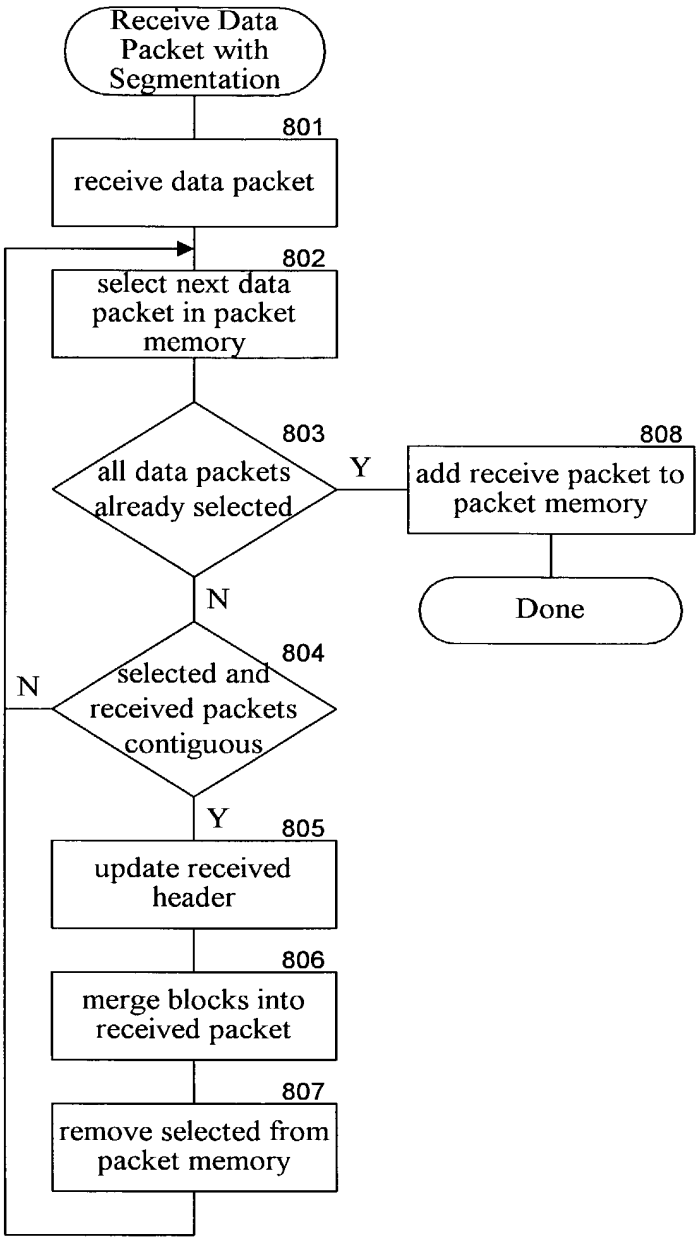
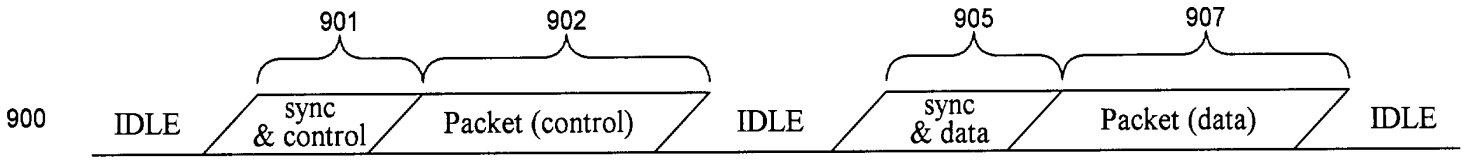

Receive Data Packet with Segmentation
801
receive data packet
802
select next data packet in packet memory
803
all data packets already selected
Y
808
add receive packet to packet memory
Done
N
804
selected and received packets contiguous
N
Y
805
update received header
806
merge blocks into received packet
807
remove selected from packet memory

Fig. 8



sync & packet type

Fig. 9A

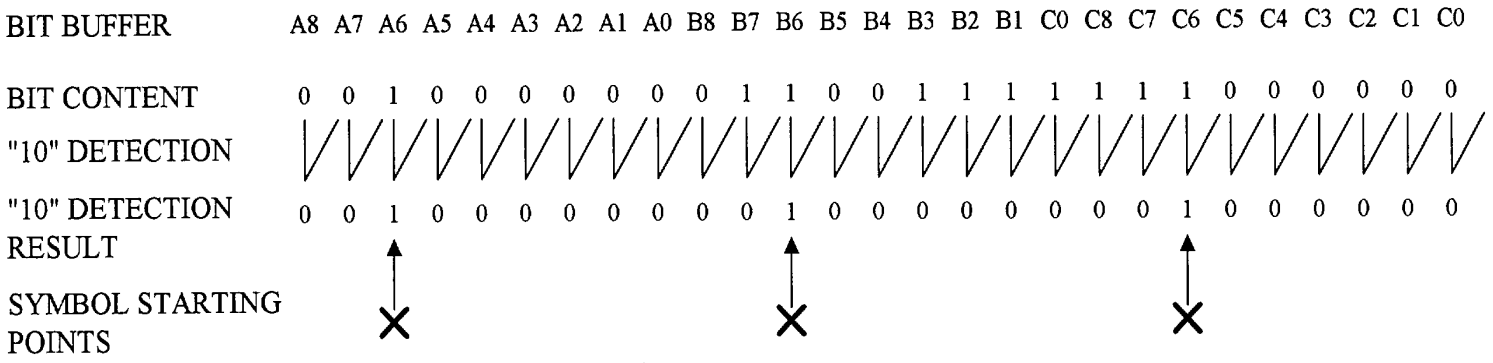

BIT BUFFER
A8 A7 A6 A5 A4 A3 A2 A1 A0 B8 B7 B6 B5 B4 B3 B2 B1 C0 C8 C7 C6 C5 C4 C3 C2 C1 C0
BIT CONTENT
0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
"10" DETECTION
"10" DETECTION RESULT
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
SYMBOL STARTING POINTS

Fig. 9B

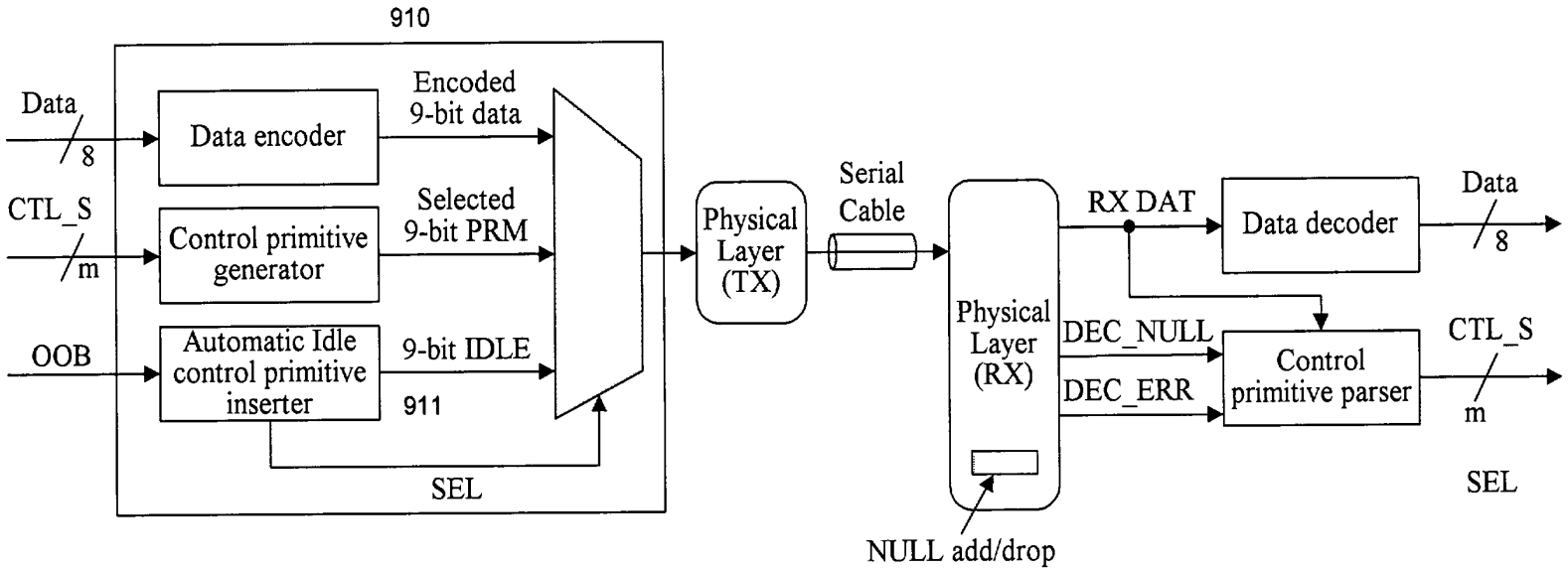

910
Data
8
CTL_S
m
OOB
Data encoder
Control primitive generator
Automatic Idle control primitive inserter
Encoded 9-bit data
Selected 9-bit PRM
9-bit IDLE
911
SEL
Physical Layer (TX)
Serial Cable
Physical Layer (RX)
NULL add/drop
RX DAT
DEC_NULL
DEC_ERR
Data decoder
Control primitive parser
Data
8
CTL_S
m
SEL

Fig. 9C

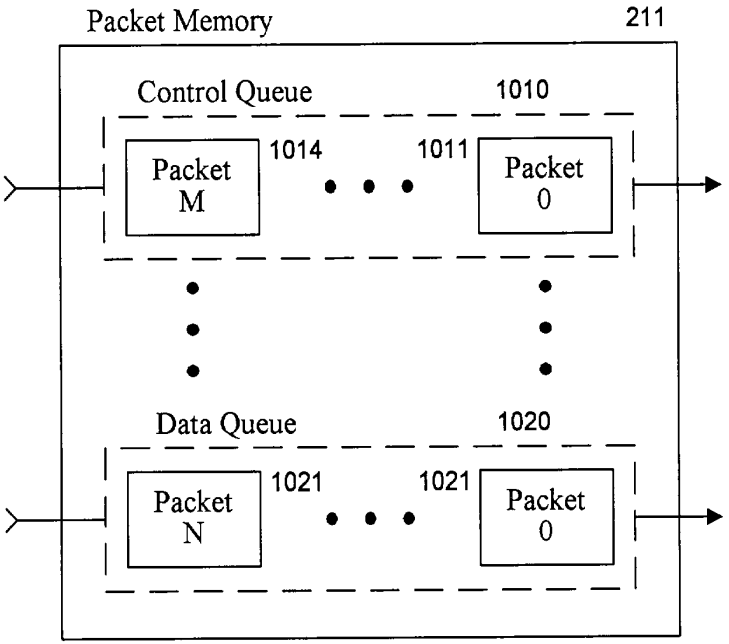

Packet Memory
211
Control Queue
1010
Packet M
1014
1011
Packet 0
Data Queue
1020
Packet N
1021
1021
Packet 0

Fig. 10

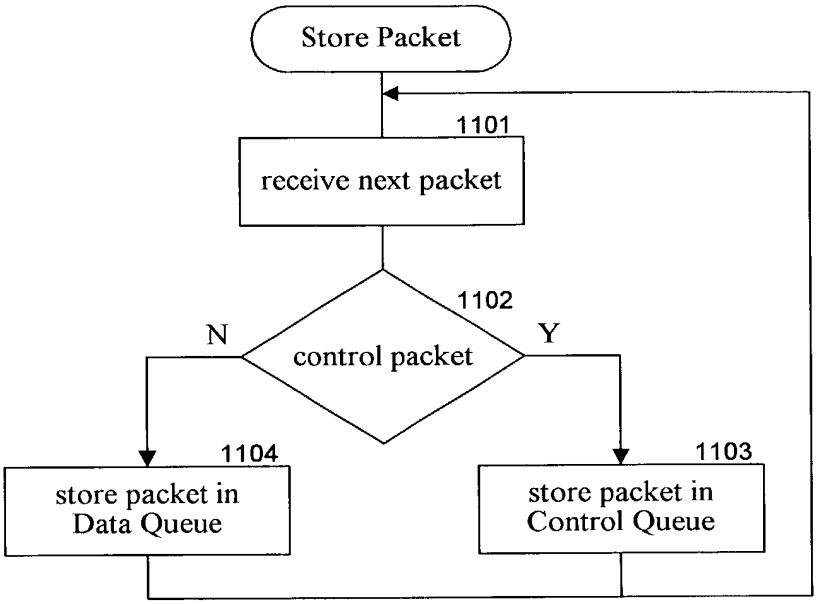

Store Packet
1101
receive next packet
1102
N
control packet
Y
1104
store packet in Data Queue
1103
store packet in Control Queue

Fig. 11

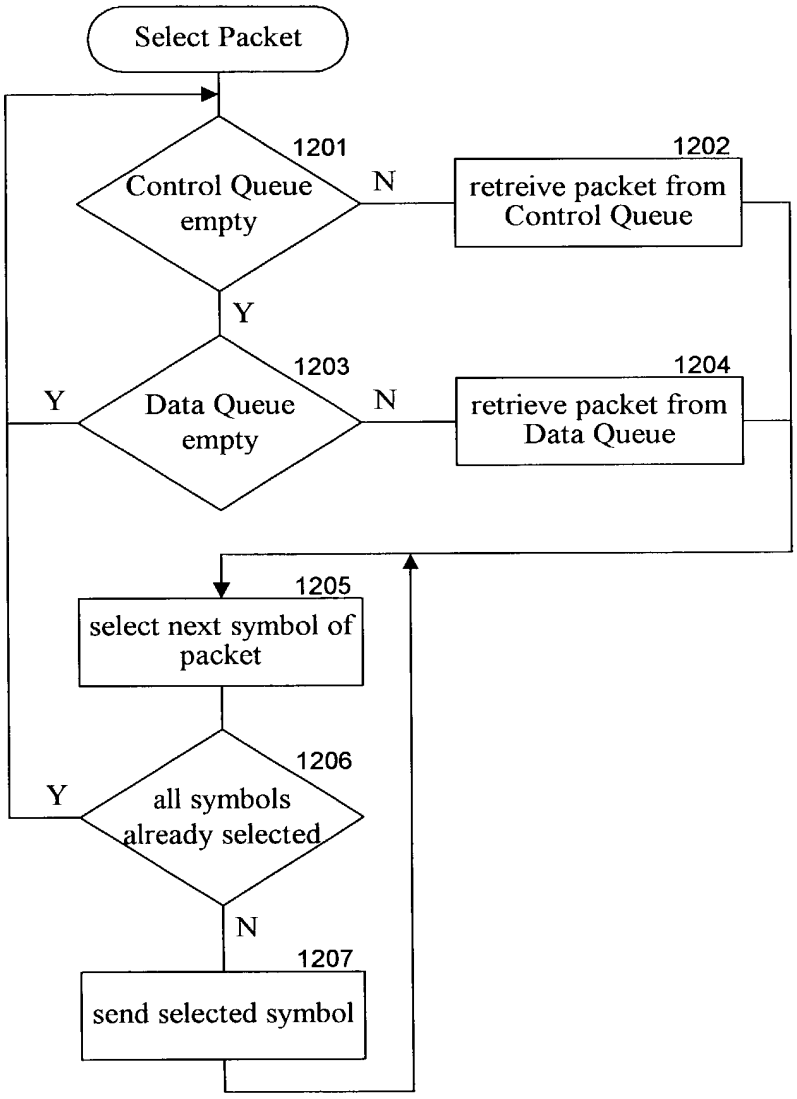

Select Packet
1201
Control Queue empty
N
1202
retreive packet from Control Queue
Y
1203
Data Queue empty
Y
N
1204
retrieve packet from Data Queue
1205
select next symbol of packet
1206
all symbols already selected
Y
N
1207
send selected symbol

Fig. 12

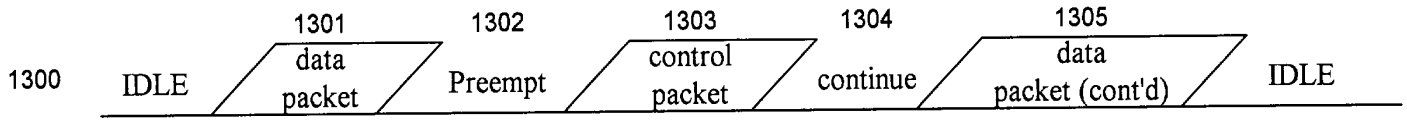

1300
IDLE
1301
data packet
1302
Preempt
1303
control packet
1304
continue
1305
data packet (cont'd)
IDLE

Fig. 13

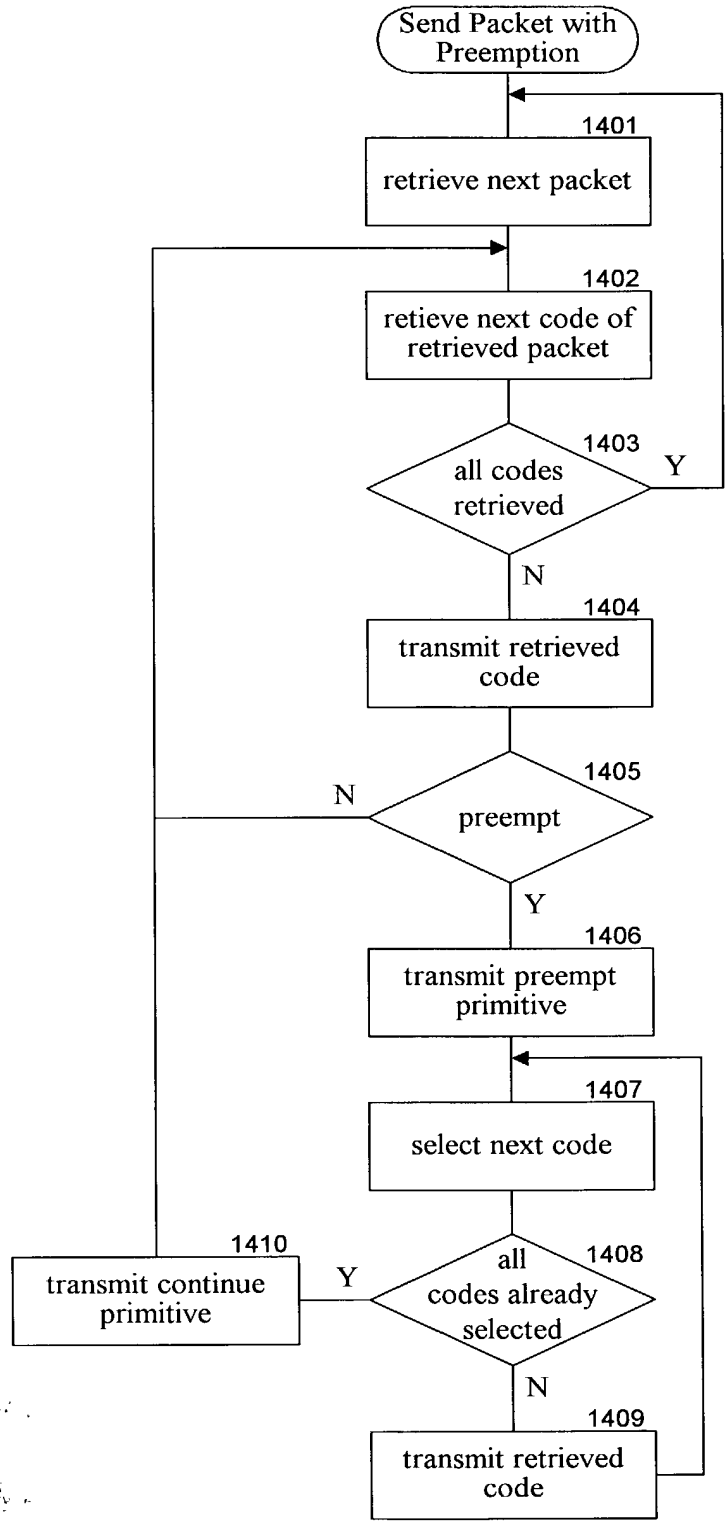

Send Packet with Preemption
1401
retrieve next packet
1402
retieve next code of retrieved packet
1403
all codes retrieved
Y
N
1404
transmit retrieved code
1405
preempt
N
Y
1406
transmit preempt primitive
1407
select next code
1408
all codes already selected
Y
1410
transmit continue primitive
N
1409
transmit retrieved code

Fig. 14

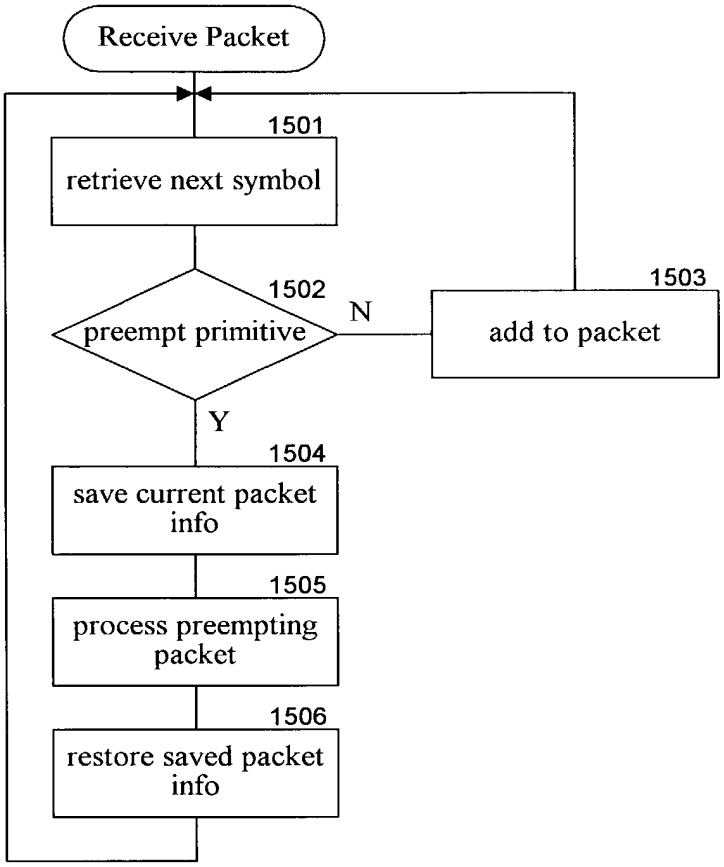

Receive Packet
1501
retrieve next symbol
1502
preempt primitive
N
1503
add to packet
Y
1504
save current packet info
1505
process preempting packet
1506
restore saved packet info

Fig. 15

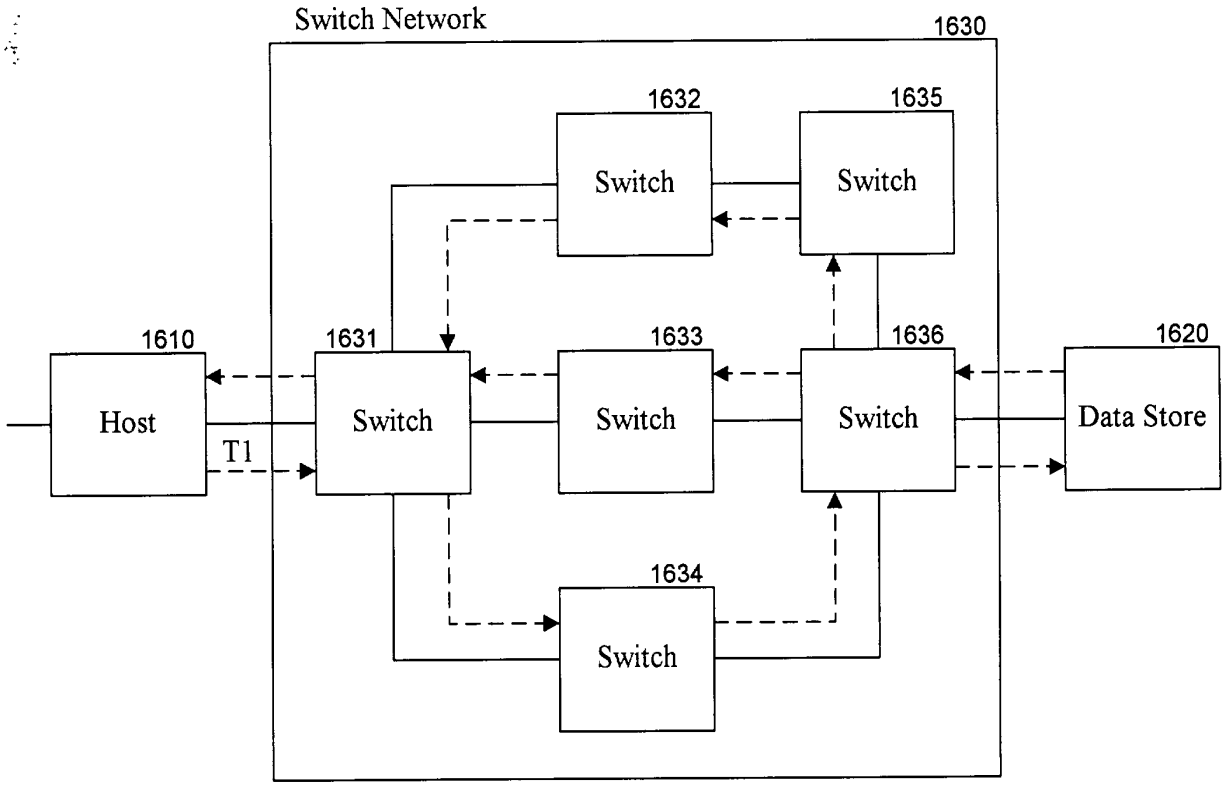

Switch Network
1630
1632
1635
Switch
Switch
1610
1631
1633
1636
1620
Host
Switch
Switch
Switch
Data Store
T1
1634
Switch

Fig. 16

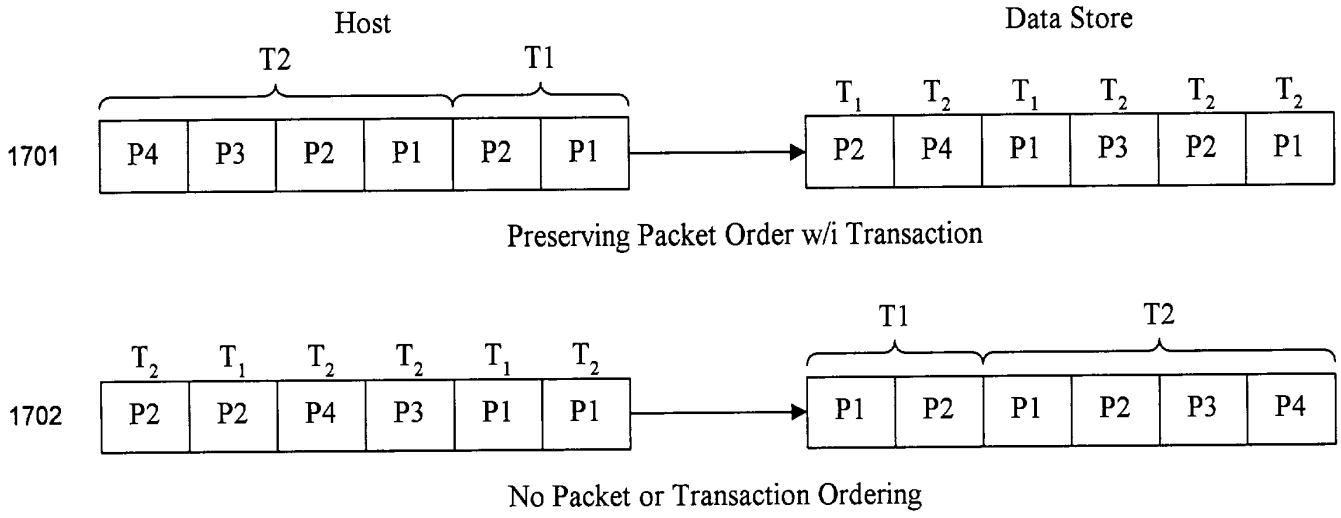

Host
Data Store
T2
T1
1701
P4
P3
P2
P1
P2
P1
T_1 T_2 T_1 T_2 T_2 T_2
P2
P4
P1
P3
P2
P1
Preserving Packet Order w/i Transaction
T1
T2
1702
T_2 T_1 T_2 T_2 T_1 T_2
P2
P2
P4
P3
P1
P1
P1
P2
P1
P2
P3
P4
No Packet or Transaction Ordering

Fig. 17

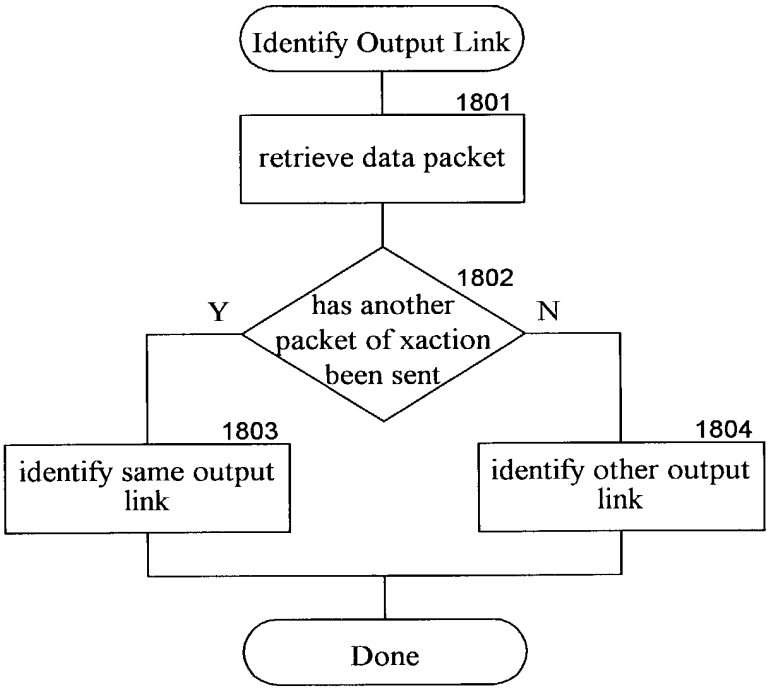

Identify Output Link
1801
retrieve data packet
1802
has another packet of xaction been sent
Y
N
1803
identify same output link
1804
identify other output link
Done

Fig. 18

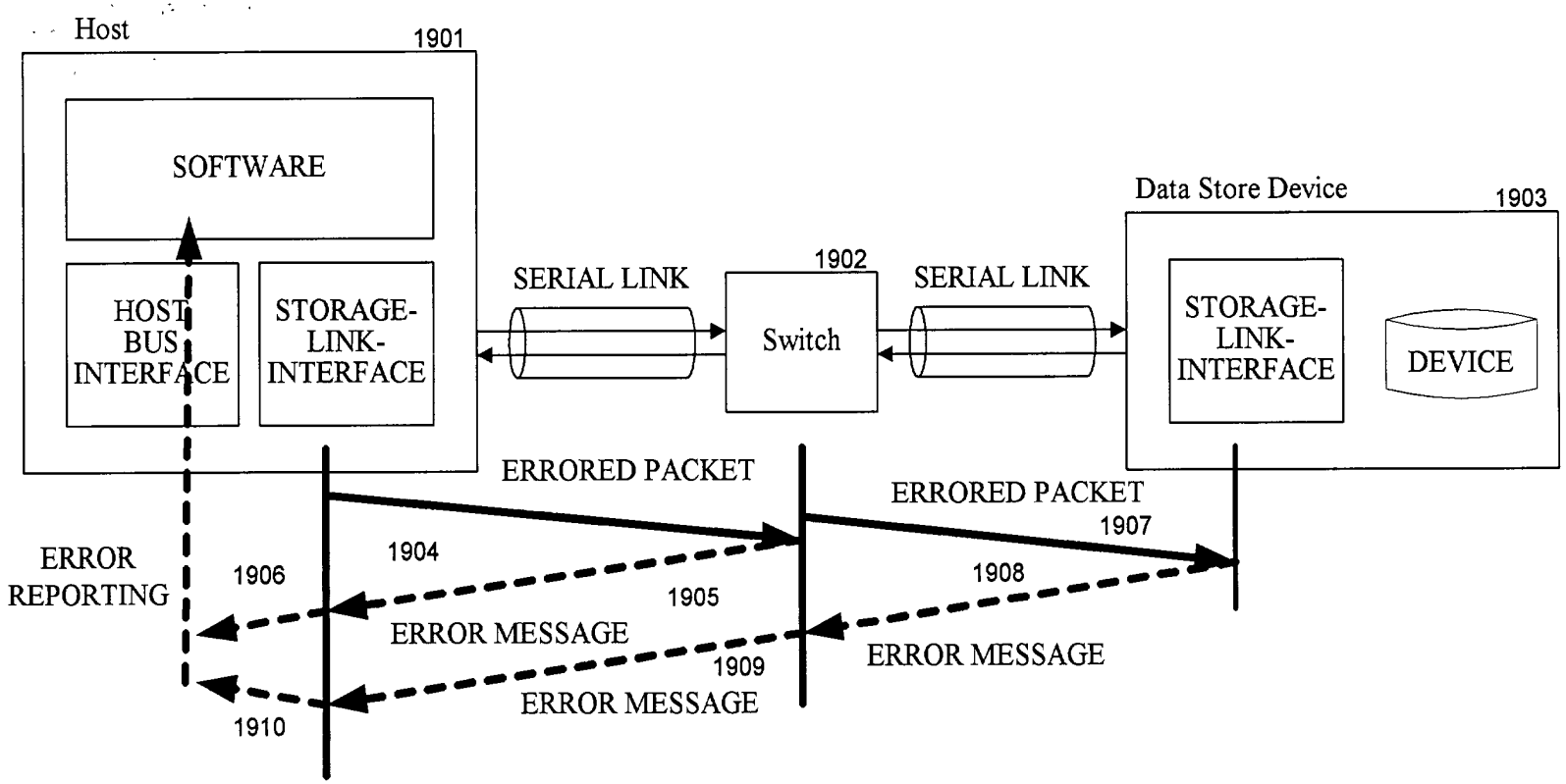

Host
1901
SOFTWARE
HOST BUS INTERFACE
STORAGE-LINK-INTERFACE
SERIAL LINK
1902
Switch
SERIAL LINK
Data Store Device
1903
STORAGE-LINK-INTERFACE
DEVICE
ERRORED PACKET
ERRORED PACKET
1907
ERROR REPORTING
1906
1904
1905
1908
ERROR MESSAGE
ERROR MESSAGE
1909
ERROR MESSAGE
1910

Fig. 19A

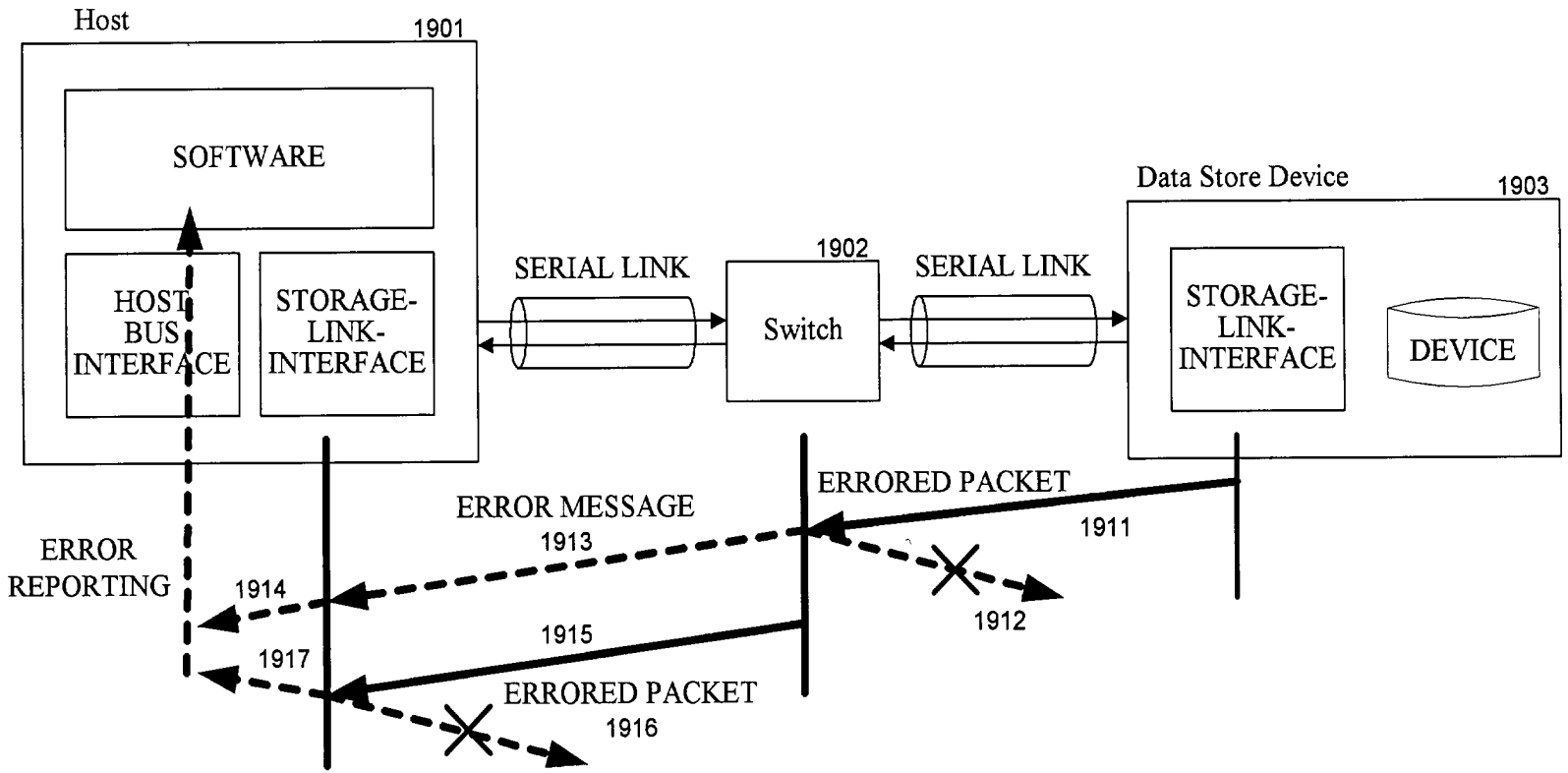

Host
1901
SOFTWARE
HOST BUS INTERFACE
STORAGE-LINK-INTERFACE
SERIAL LINK
1902
Switch
SERIAL LINK
Data Store Device
1903
STORAGE-LINK-INTERFACE
DEVICE
ERRORED PACKET
1911
ERROR MESSAGE
1913
ERROR REPORTING
1914
1912
1915
1917
ERRORED PACKET
1916

Fig. 19B

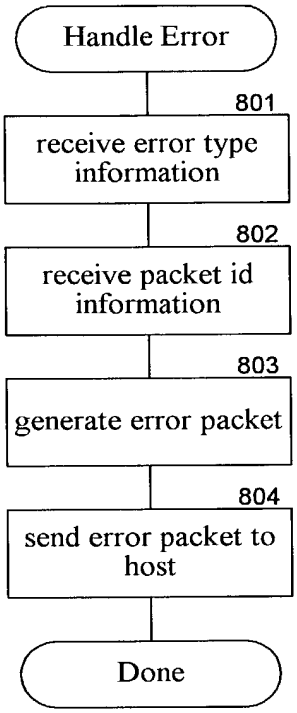

Handle Error
801
receive error type information
802
receive packet id information
803
generate error packet
804
send error packet to host
Done

Fig. 19C

8b code	9 bit symbol
0000 0000	101010101
0000 0001	101010100
0000 0010	101010111
⋮	
0101 0101	001010101
⋮	
0111 0110	001110110
0111 0111	100100010
⋮	
1111 1111	110101010

Fig. 20

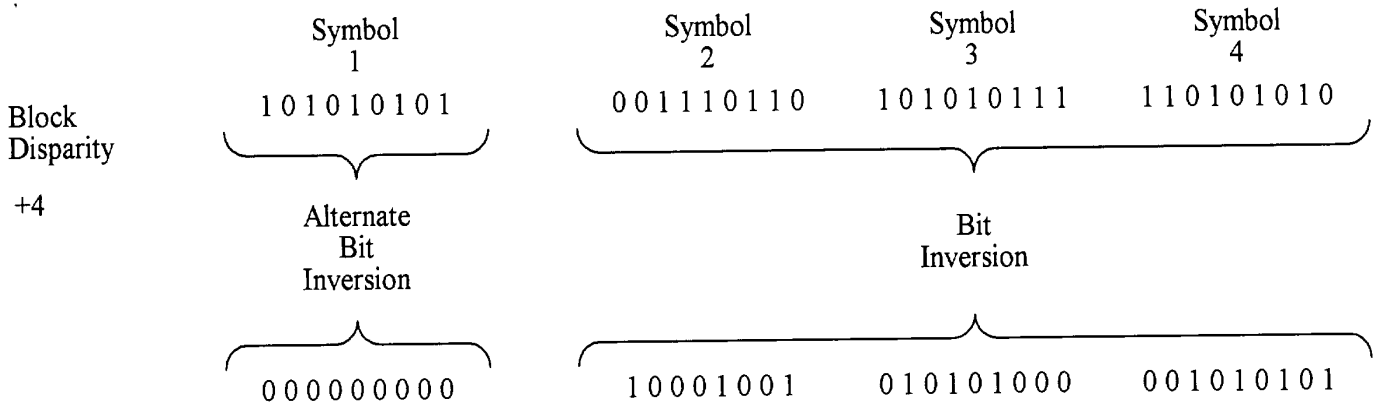

Block Disparity
+4
Symbol 1
101010101
Symbol 2
001110110
Symbol 3
101010111
Symbol 4
110101010
Alternate Bit Inversion
000000000
Bit Inversion
10001001
010101000
001010101

Fig. 21A

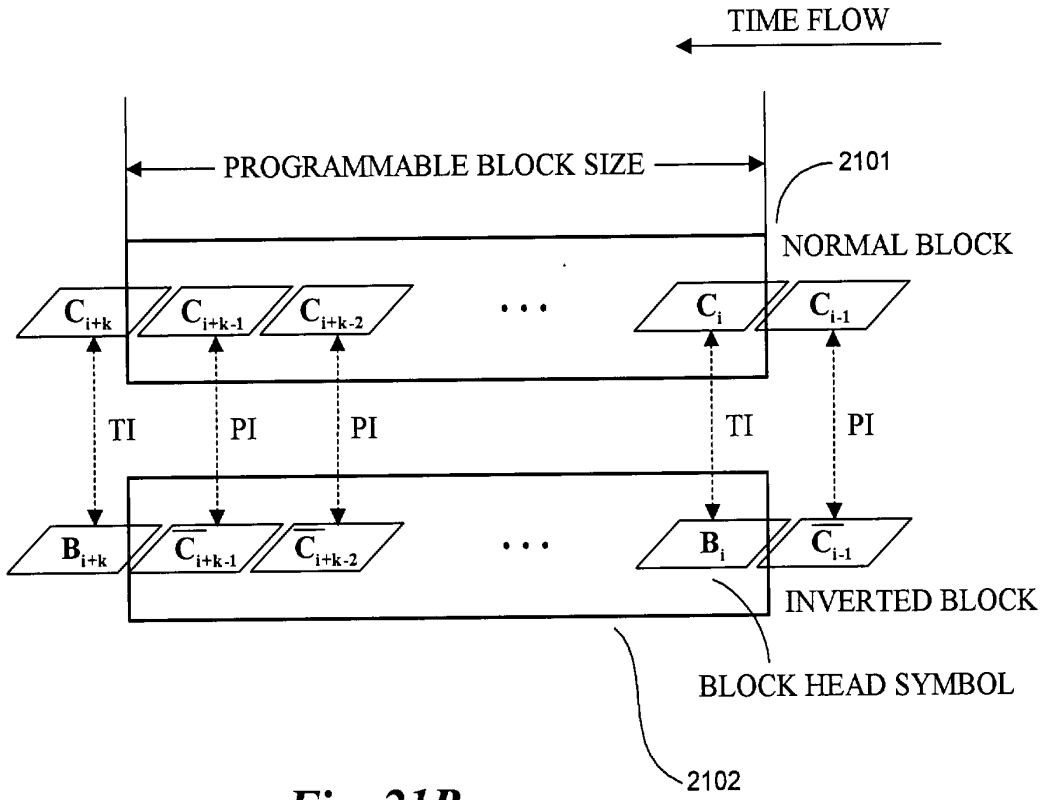

TIME FLOW
PROGRAMMABLE BLOCK SIZE
2101
NORMAL BLOCK
C_{i+k}
C_{i+k-1}
C_{i+k-2}
C_i
C_{i-1}
TI
PI
PI
TI
PI
B_{i+k}
$\overline{C}_{i+k-1}$
$\overline{C}_{i+k-2}$
B_i
$\overline{C}_{i-1}$
INVERTED BLOCK
BLOCK HEAD SYMBOL
2102

Fig. 21B

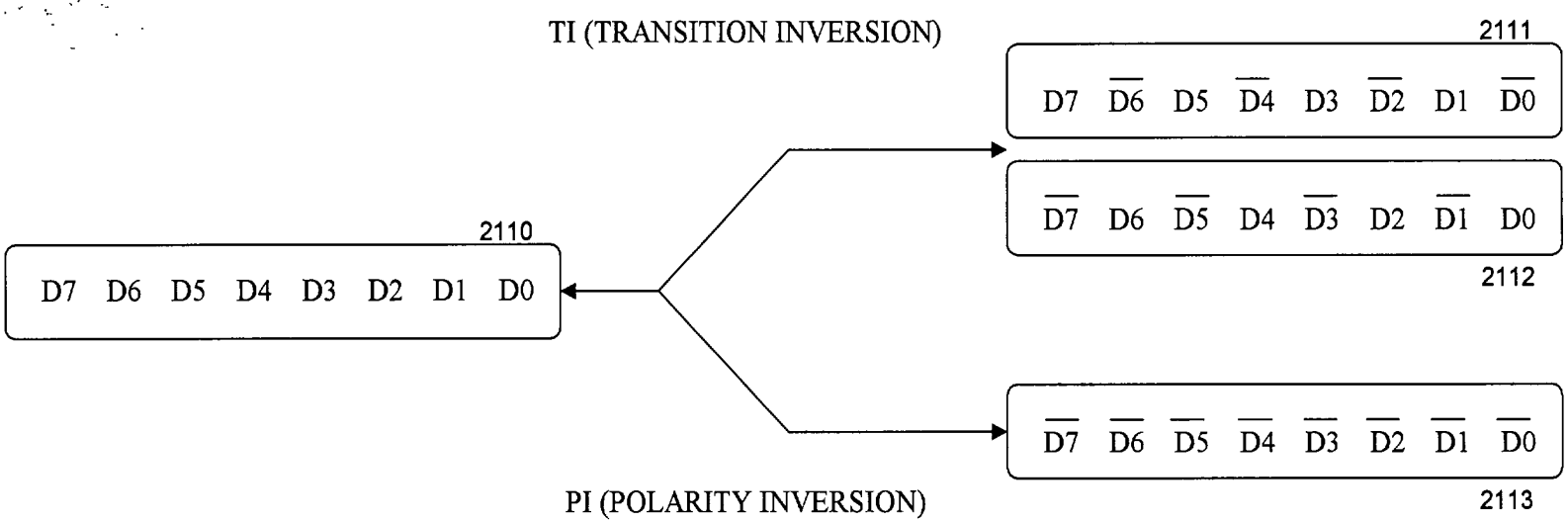

TI (TRANSITION INVERSION)
2111
D7 $\overline{D6}$ D5 $\overline{D4}$ D3 $\overline{D2}$ D1 $\overline{D0}$
$\overline{D7}$ D6 $\overline{D5}$ D4 $\overline{D3}$ D2 $\overline{D1}$ D0
2112
2110
D7 D6 D5 D4 D3 D2 D1 D0
$\overline{D7}$ $\overline{D6}$ $\overline{D5}$ $\overline{D4}$ $\overline{D3}$ $\overline{D2}$ $\overline{D1}$ $\overline{D0}$
PI (POLARITY INVERSION)
2113

Fig. 21C

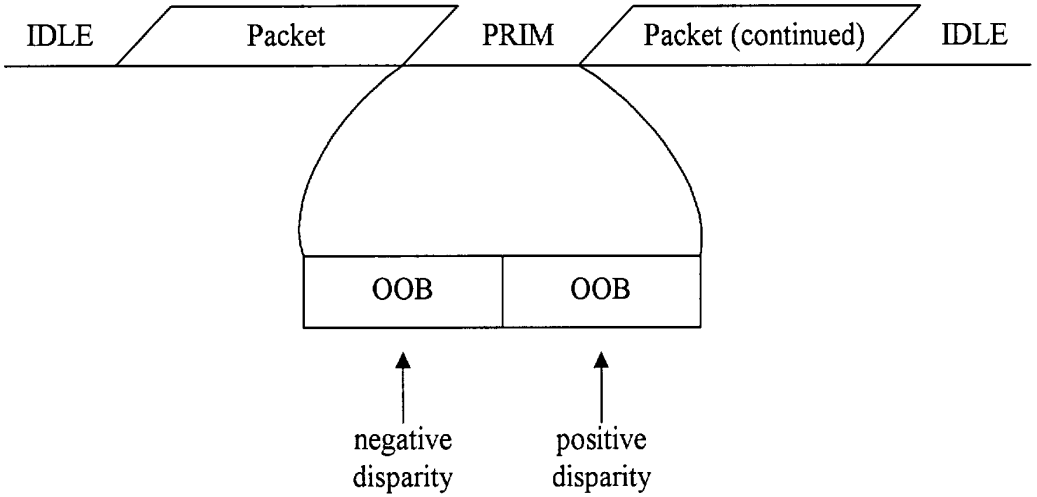

IDLE
Packet
PRIM
Packet (continued)
IDLE
OOB
OOB
negative disparity
positive disparity

Fig. 22

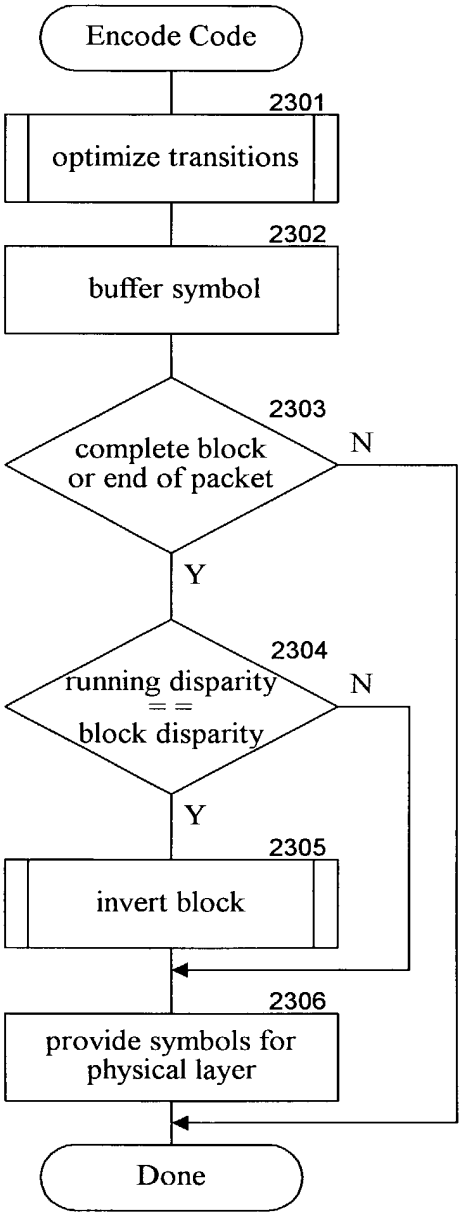

Encode Code
2301
optimize transitions
2302
buffer symbol
2303
complete block or end of packet
N
Y
2304
running disparity == block disparity
N
Y
2305
invert block
2306
provide symbols for physical layer
Done

Fig. 23

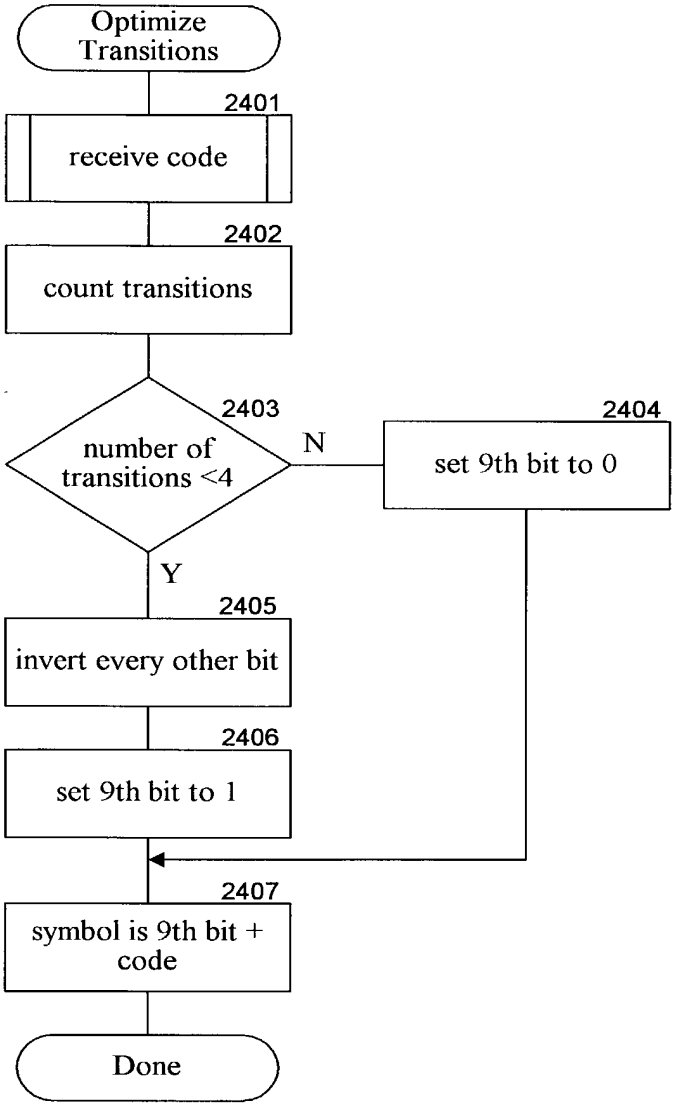

Optimize Transitions
2401
receive code
2402
count transitions
2403
number of transitions <4
N
2404
set 9th bit to 0
Y
2405
invert every other bit
2406
set 9th bit to 1
2407
symbol is 9th bit + code
Done

Fig. 24

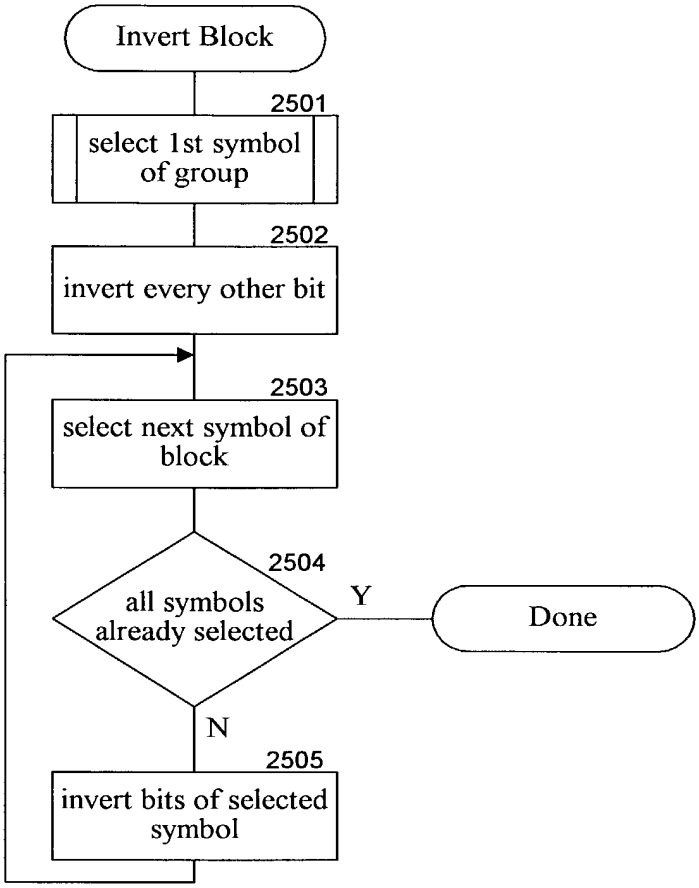

Invert Block
2501
select 1st symbol of group
2502
invert every other bit
2503
select next symbol of block
2504
all symbols already selected
Y
Done
N
2505
invert bits of selected symbol

Fig. 25

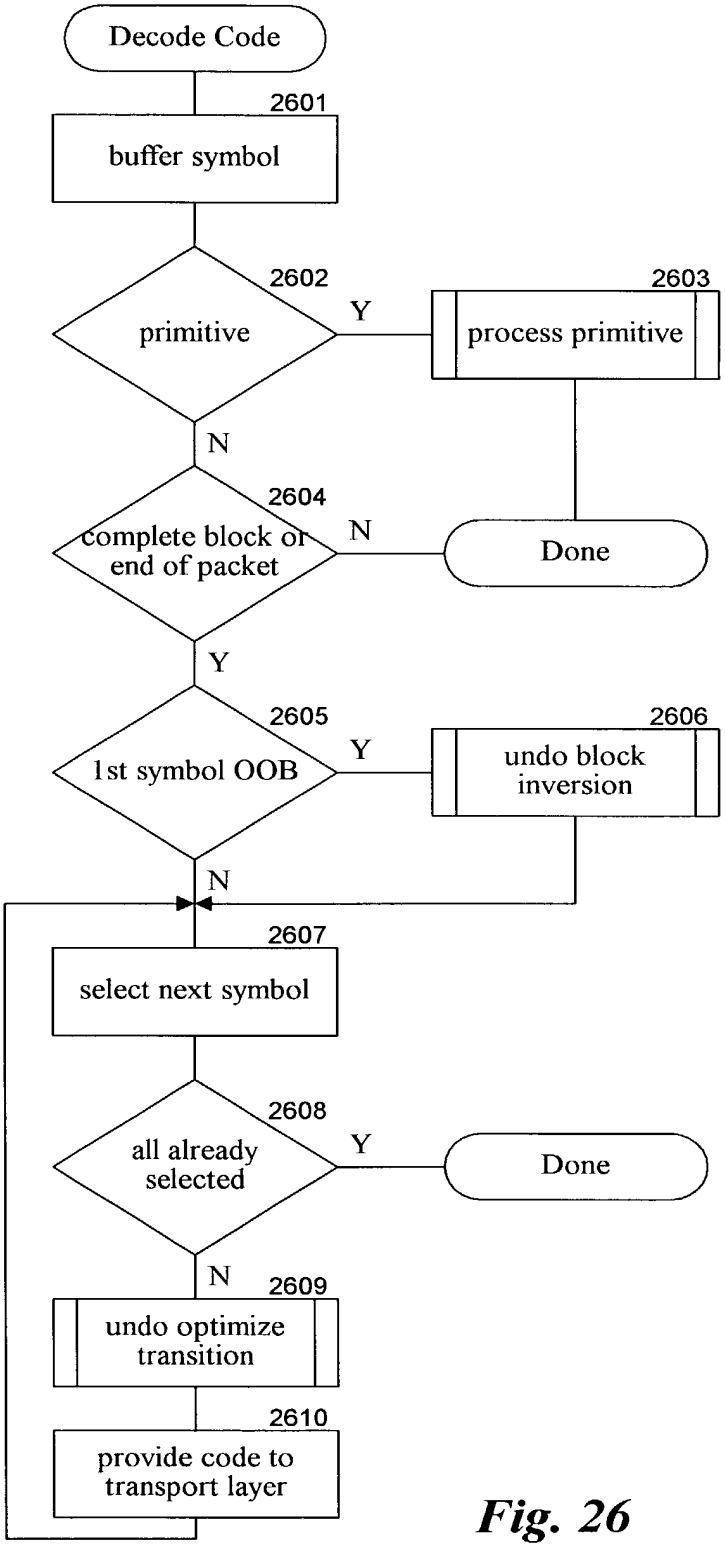

Decode Code
2601
buffer symbol
2602
primitive
Y
2603
process primitive
N
2604
complete block or end of packet
N
Done
Y
2605
1st symbol OOB
Y
2606
undo block inversion
N
2607
select next symbol
2608
all already selected
Y
Done
N
2609
undo optimize transition
2610
provide code to transport layer

Fig. 26

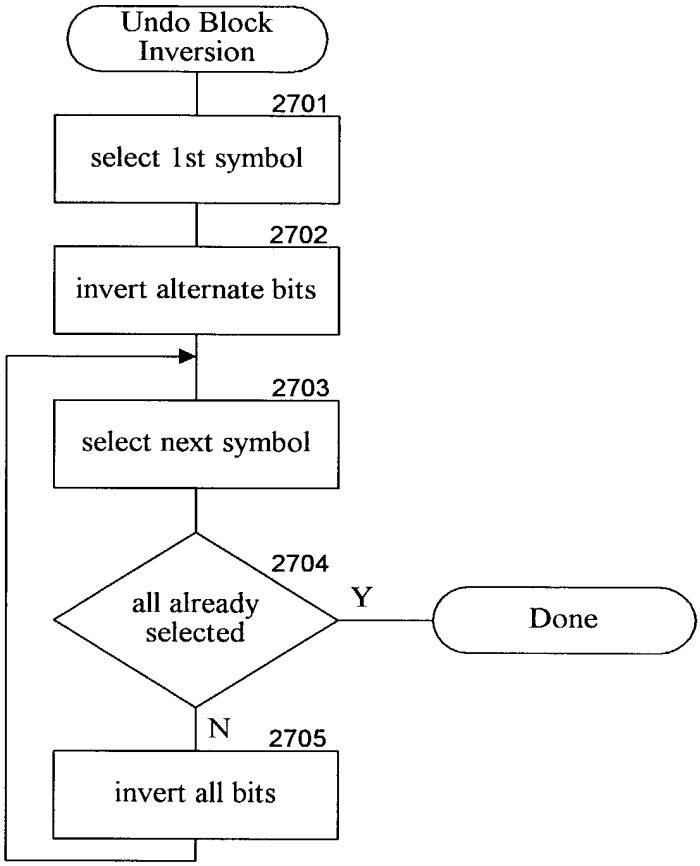

Undo Block Inversion
2701
select 1st symbol
2702
invert alternate bits
2703
select next symbol
2704
all already selected
Y
Done
N
2705
invert all bits

Fig. 27

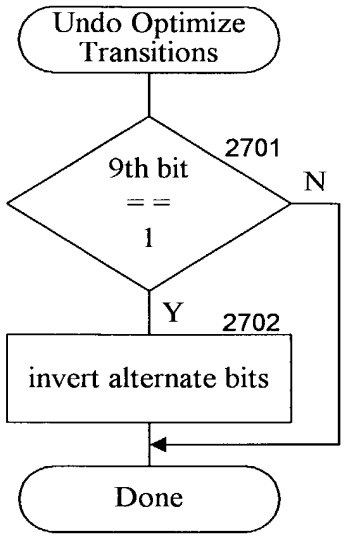

Undo Optimize Transitions
9th bit == 1
2701
N
Y
2702
invert alternate bits
Done

Fig. 28

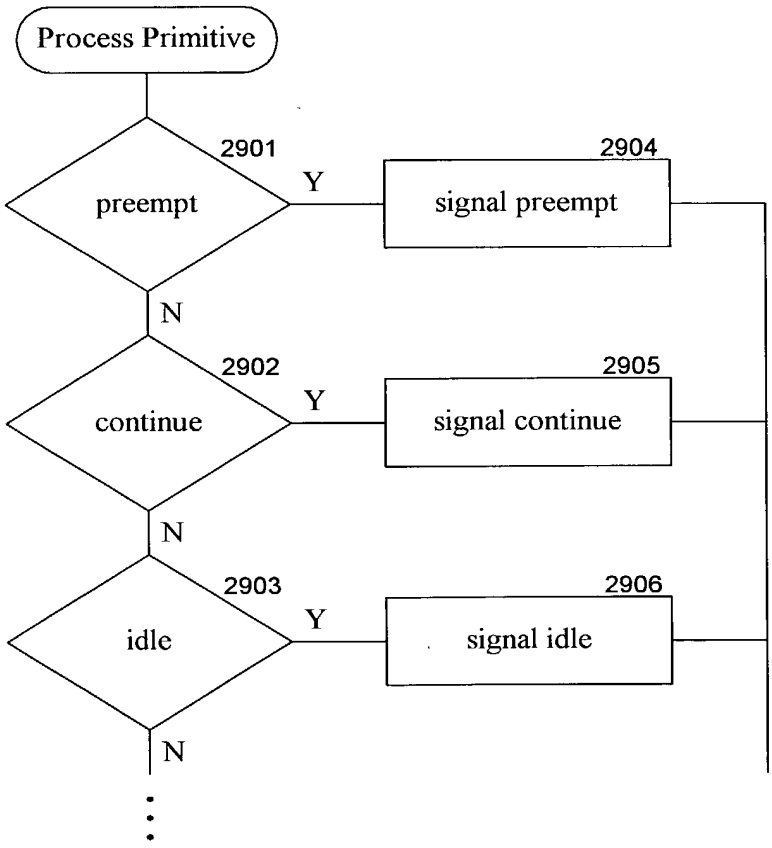

Process Primitive
2901
preempt
Y
2904
signal preempt
N
2902
continue
Y
2905
signal continue
N
2903
idle
Y
2906
signal idle
N

Fig. 29

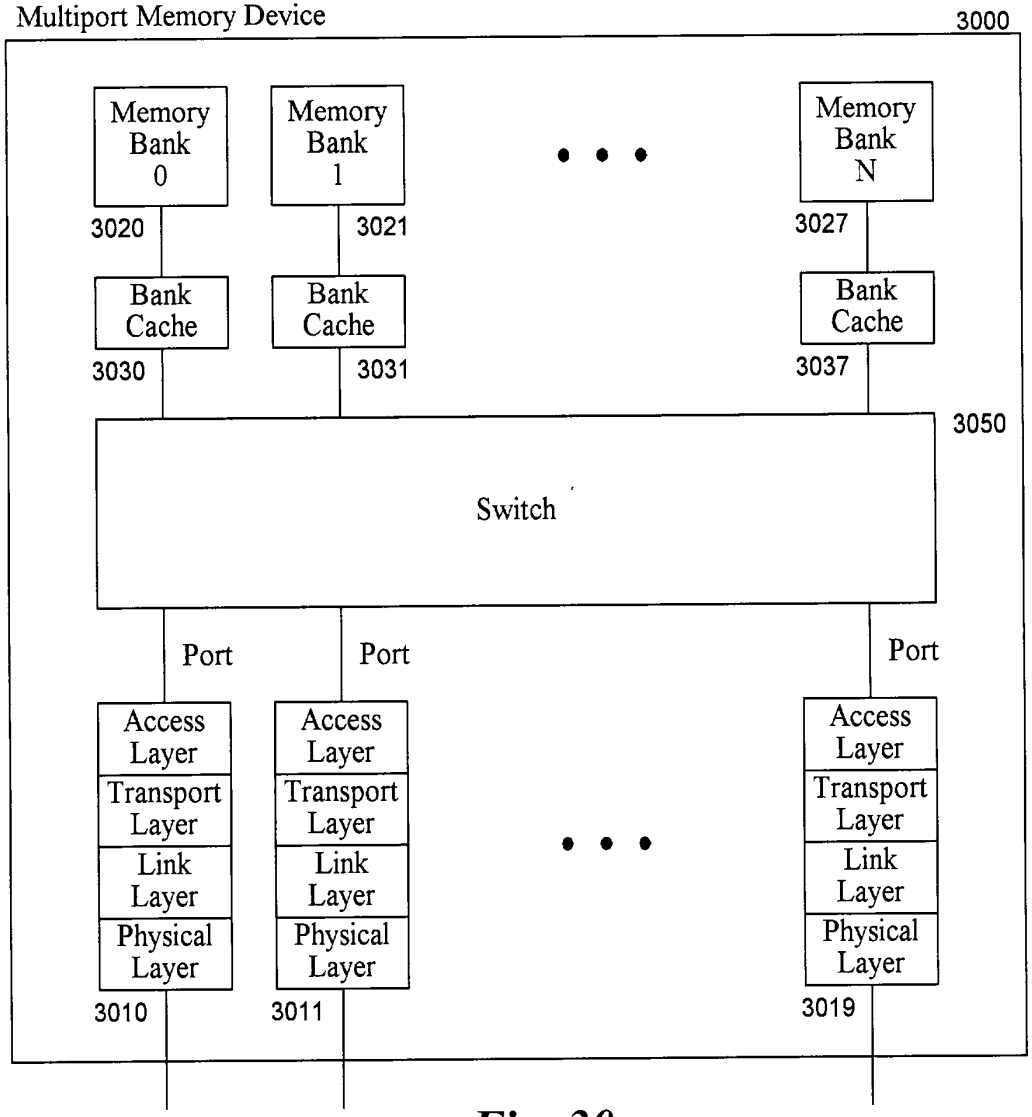

Multiport Memory Device
3000
Memory Bank 0
Memory Bank 1
Memory Bank N
3020
3021
3027
Bank Cache
Bank Cache
Bank Cache
3030
3031
3037
3050
Switch
Port
Port
Port
Access Layer
Transport Layer
Link Layer
Physical Layer
Access Layer
Transport Layer
Link Layer
Physical Layer
Access Layer
Transport Layer
Link Layer
Physical Layer
3010
3011
3019

Fig. 30

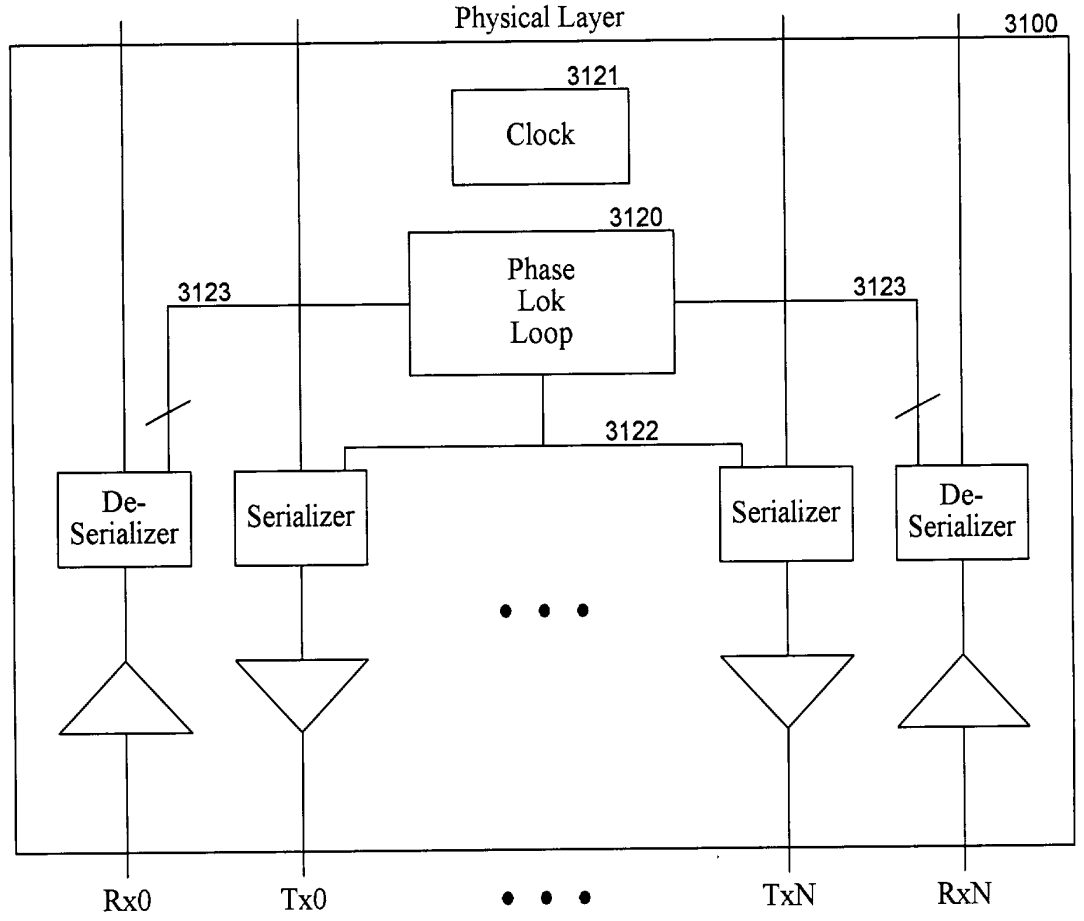

Physical Layer
3100
3121
Clock
3120
Phase
Lok
Loop
3123
3123
3122
De-
Serializer
Serializer
Serializer
De-
Serializer
Rx0
Tx0
TxN
RxN

Fig. 31

Input Queue 3201

Port	R/W	Address	Data
3	R	1000	
4	W	4000	10....1
3	W	1000	111...0
3	R	2000	
	⋮		

Output Queue 3202

Valid	Port	Data
1	3	11...0
0		
0		
1	3	101...1
	⋮	

Fig. 32

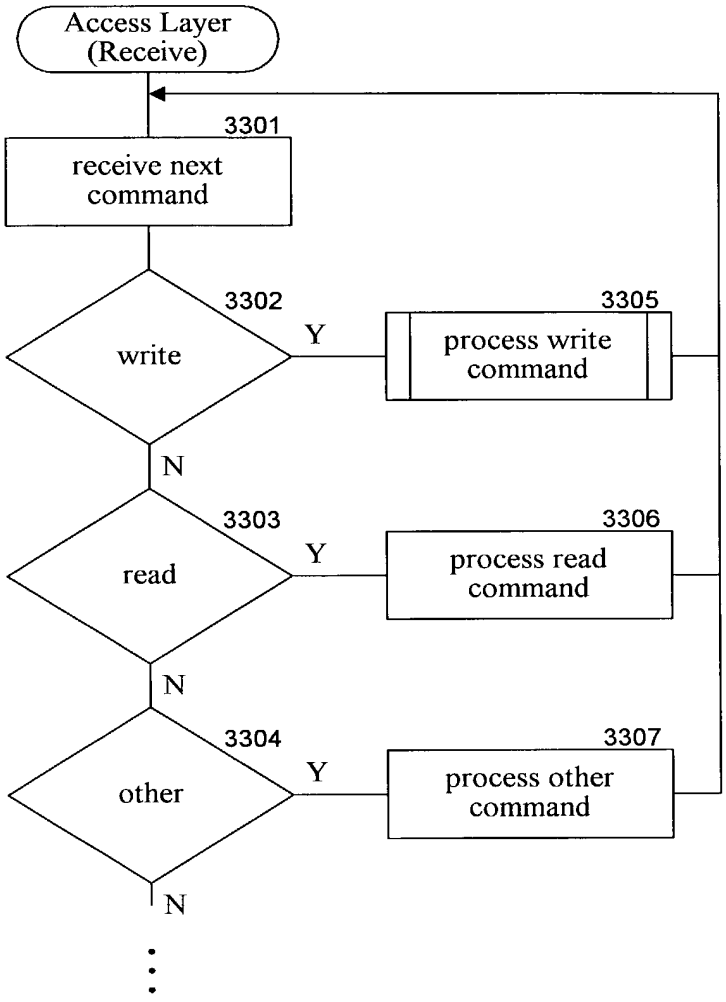

Access Layer
(Receive)
3301
receive next
command
3302
write
Y
3305
process write
command
N
3303
read
Y
3306
process read
command
N
3304
other
Y
3307
process other
command
N

Fig. 33

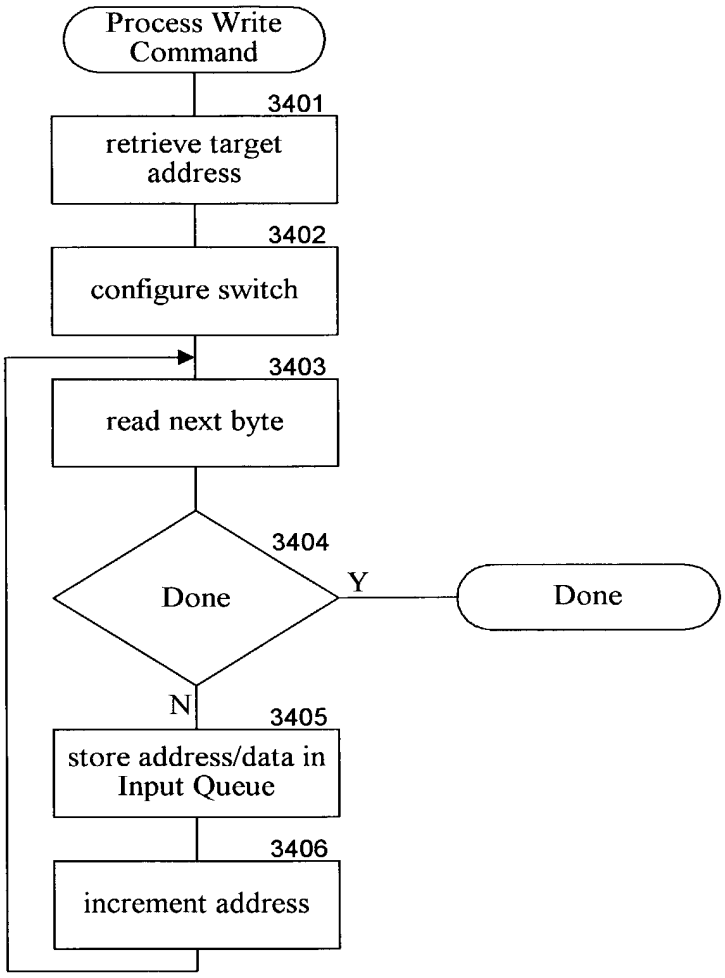

Process Write Command
3401
retrieve target address
3402
configure switch
3403
read next byte
3404
Done
Y
Done
N
3405
store address/data in Input Queue
3406
increment address

Fig. 34

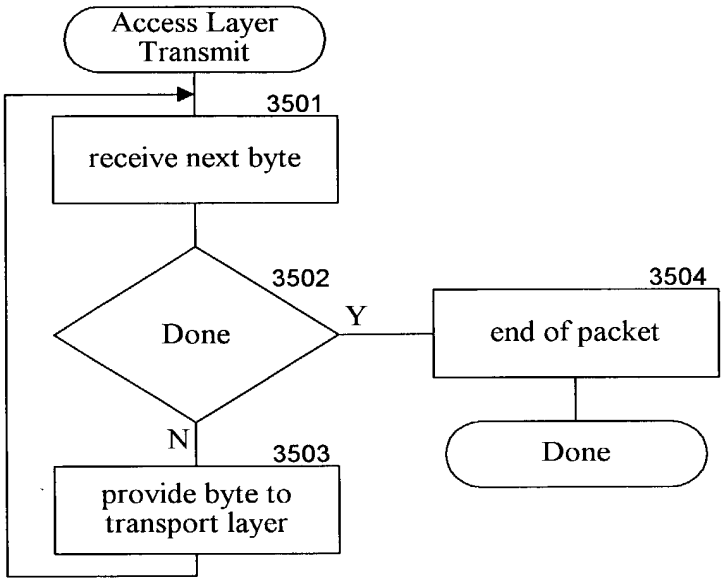

Access Layer Transmit
3501
receive next byte
3502
Done
Y
3504
end of packet
Done
N
3503
provide byte to transport layer

Fig. 35

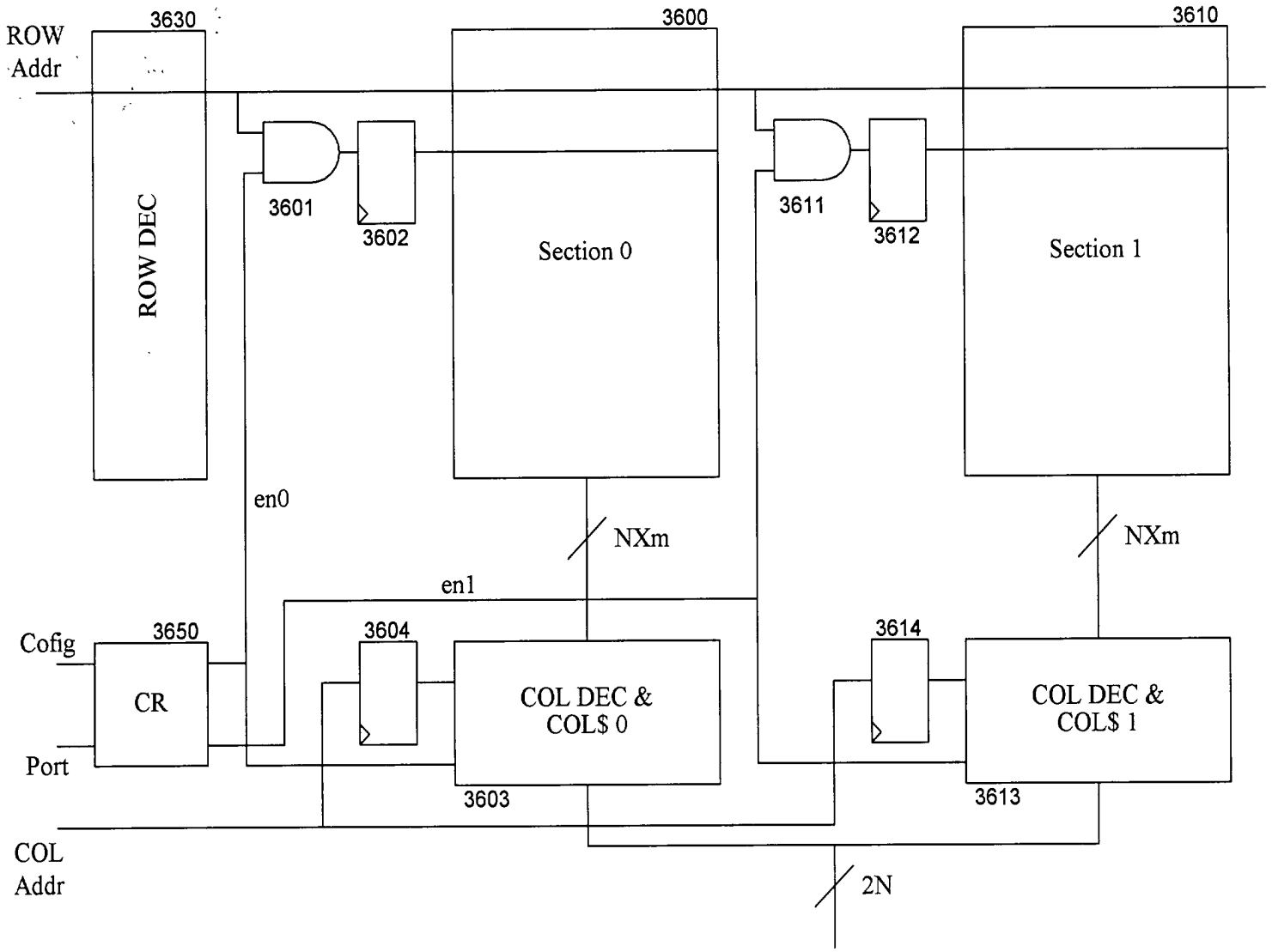
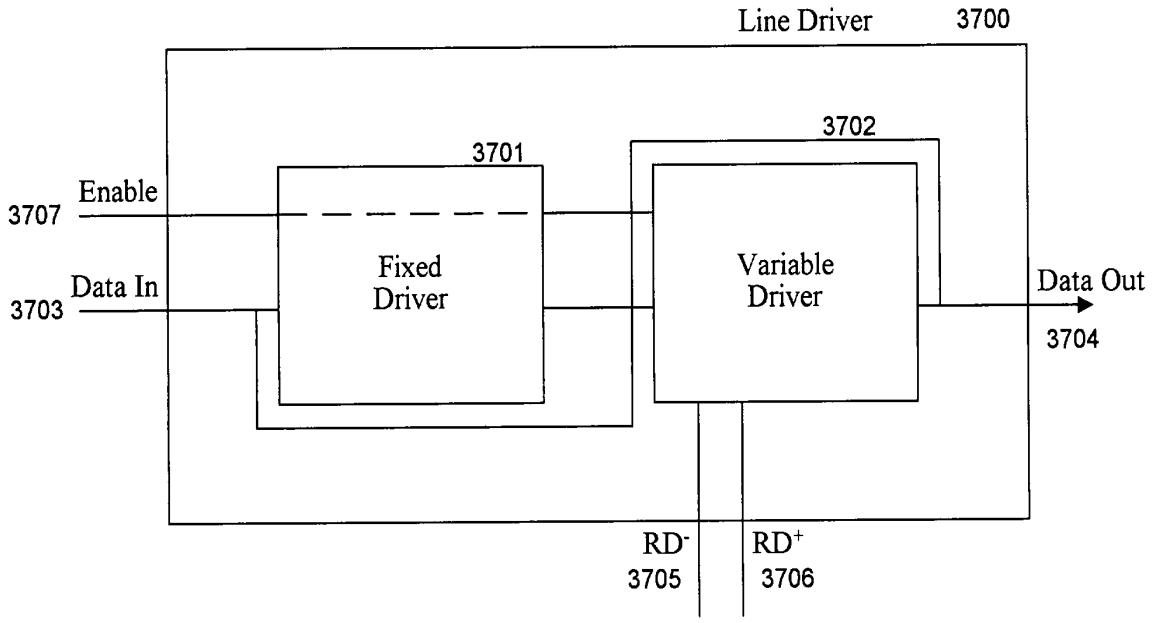

ROW
Addr
3630
ROW DEC
3600
Section 0
3610
Section 1
3601
3602
3611
3612
en0
en1
NXm
NXm
Cofig
Port
3650
CR
3604
COL DEC &
COL$ 0
3603
3614
COL DEC &
COL$ 1
3613
COL
Addr
2N

Fig. 36



Variable Driver

$\begin{cases} RD^+ \wedge \overline{DataIn} = \text{pull down} \\ RD^- \wedge DataIn = \text{pull up} \end{cases}$

Fig. 37A

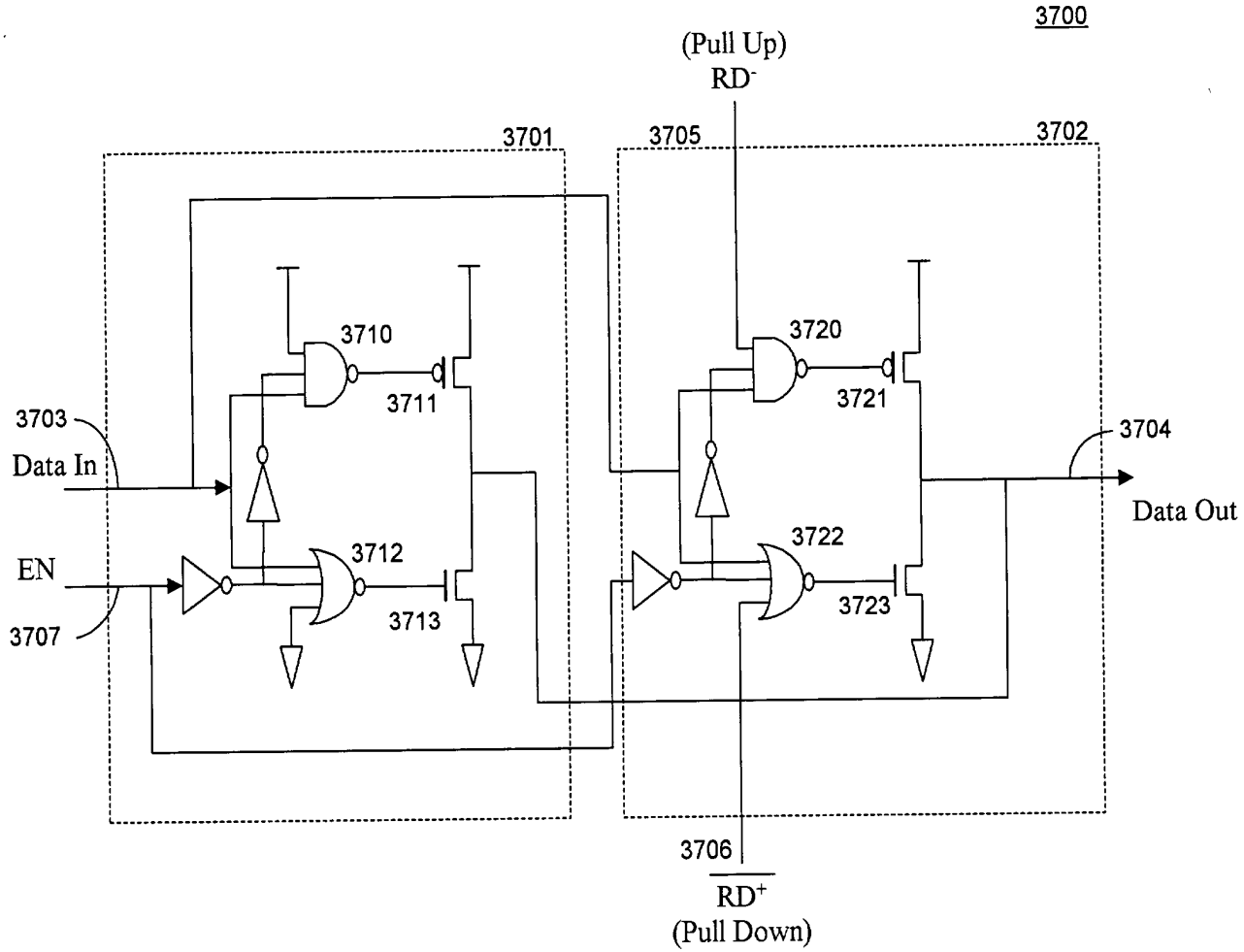

3700
(Pull Up)
RD-
3701
3705
3702
3710
3711
3720
3721
3703
Data In
3704
Data Out
3712
3722
EN
3713
3723
3707
3706
RD+
(Pull Down)

Fig. 37B

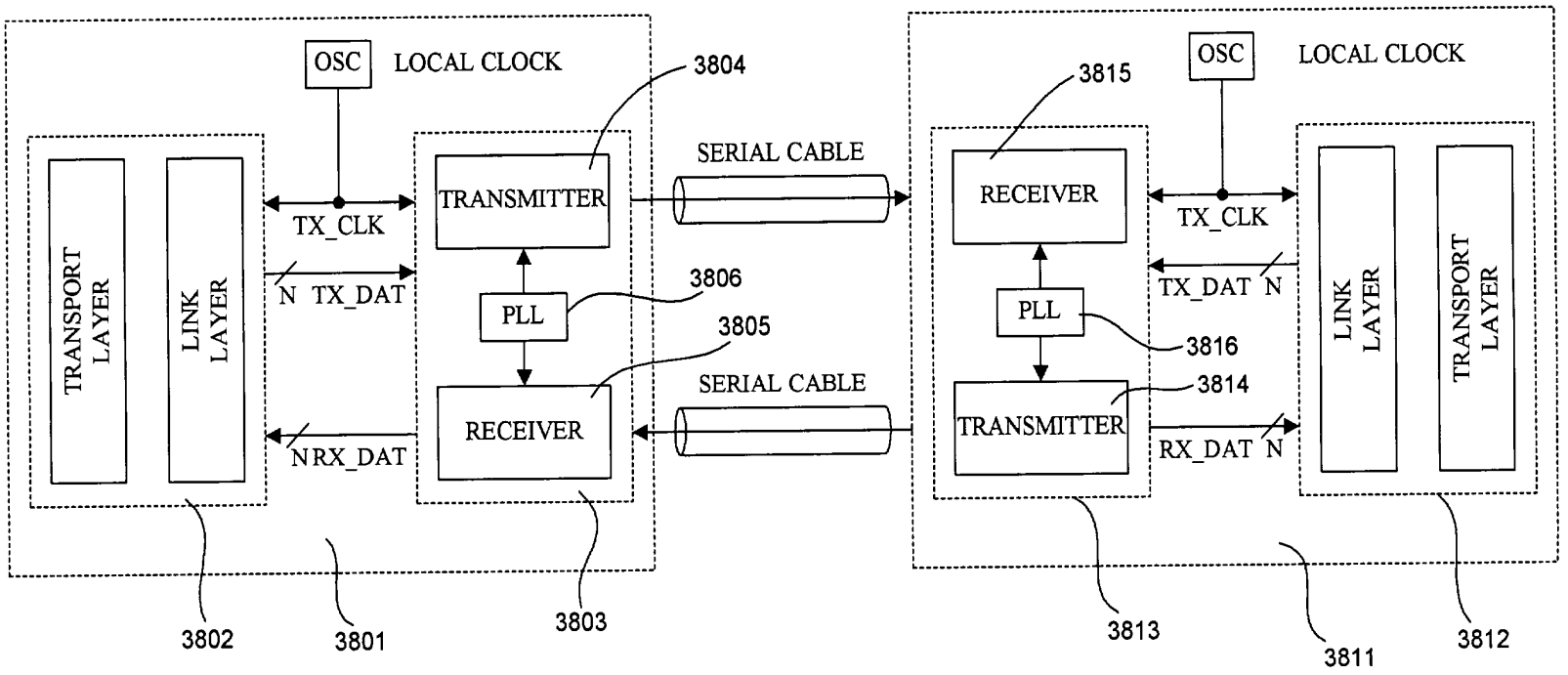

OSC
LOCAL CLOCK
3804
SERIAL CABLE
TRANSMITTER
TX_CLK
N TX_DAT
PLL
3806
3805
RECEIVER
NRX_DAT
SERIAL CABLE
TRANSPORT LAYER
LINK LAYER
3802
3801
3803
3815
OSC
LOCAL CLOCK
RECEIVER
TX_CLK
TX_DAT N
PLL
3816
3814
TRANSMITTER
RX_DAT N
LINK LAYER
TRANSPORT LAYER
3813
3811
3812

Fig. 38A

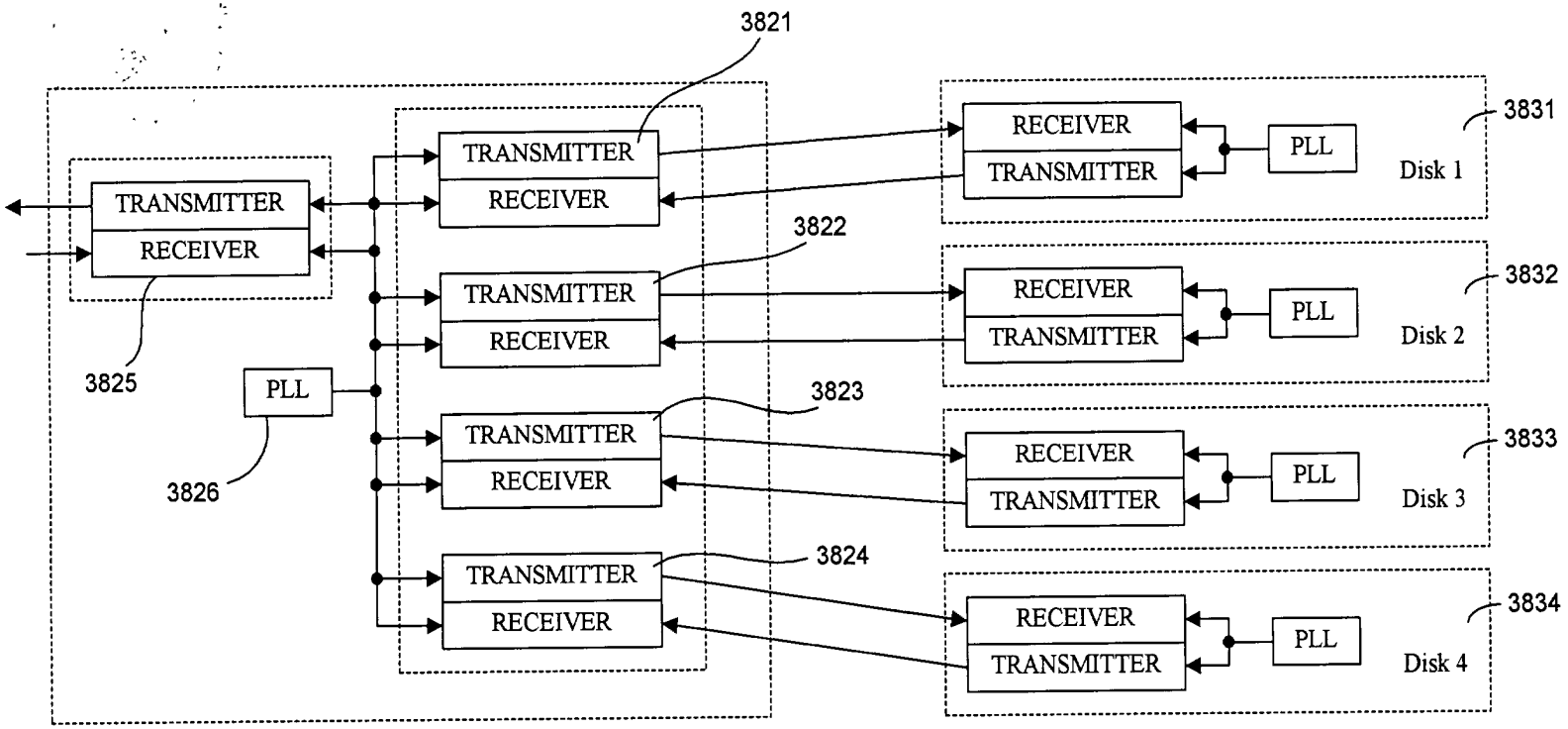

3821
TRANSMITTER
RECEIVER
TRANSMITTER
RECEIVER
3825
PLL
3826
3822
TRANSMITTER
RECEIVER
3823
TRANSMITTER
RECEIVER
3824
TRANSMITTER
RECEIVER
RECEIVER
TRANSMITTER
PLL
Disk 1
3831
RECEIVER
TRANSMITTER
PLL
Disk 2
3832
RECEIVER
TRANSMITTER
PLL
Disk 3
3833
RECEIVER
TRANSMITTER
PLL
Disk 4
3834

Fig. 38B

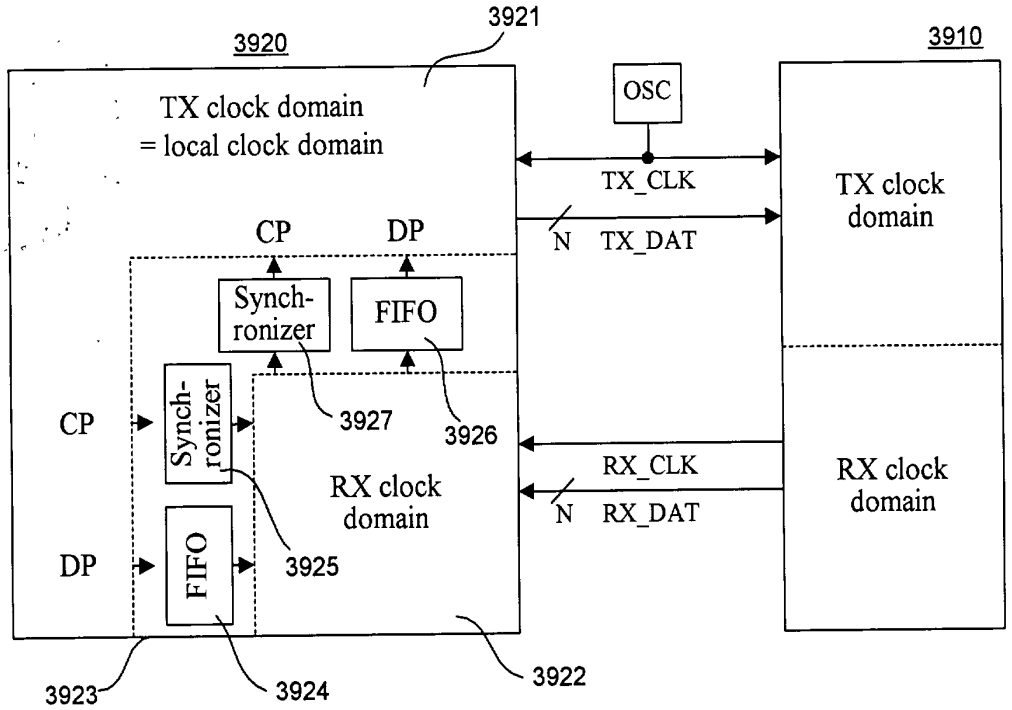


Fig. 39A

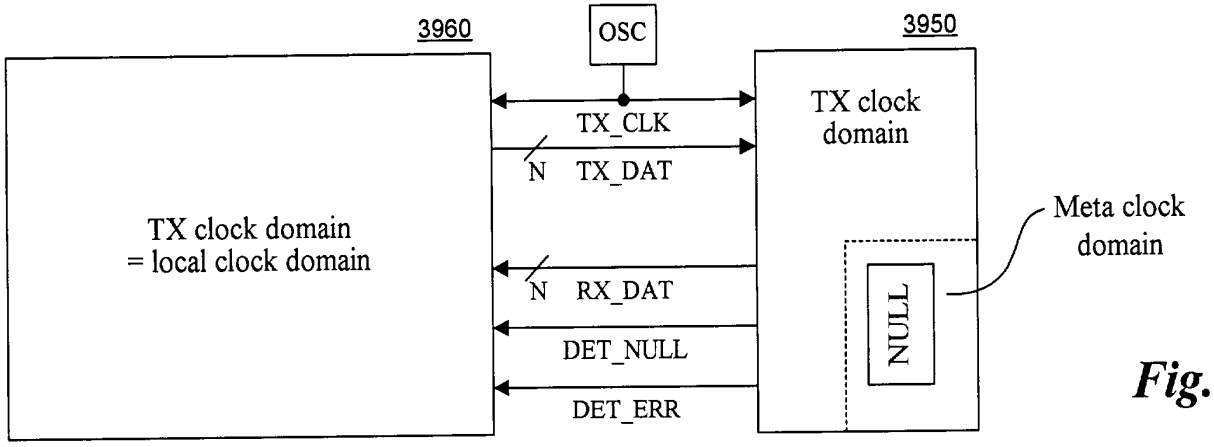


Fig. 39B

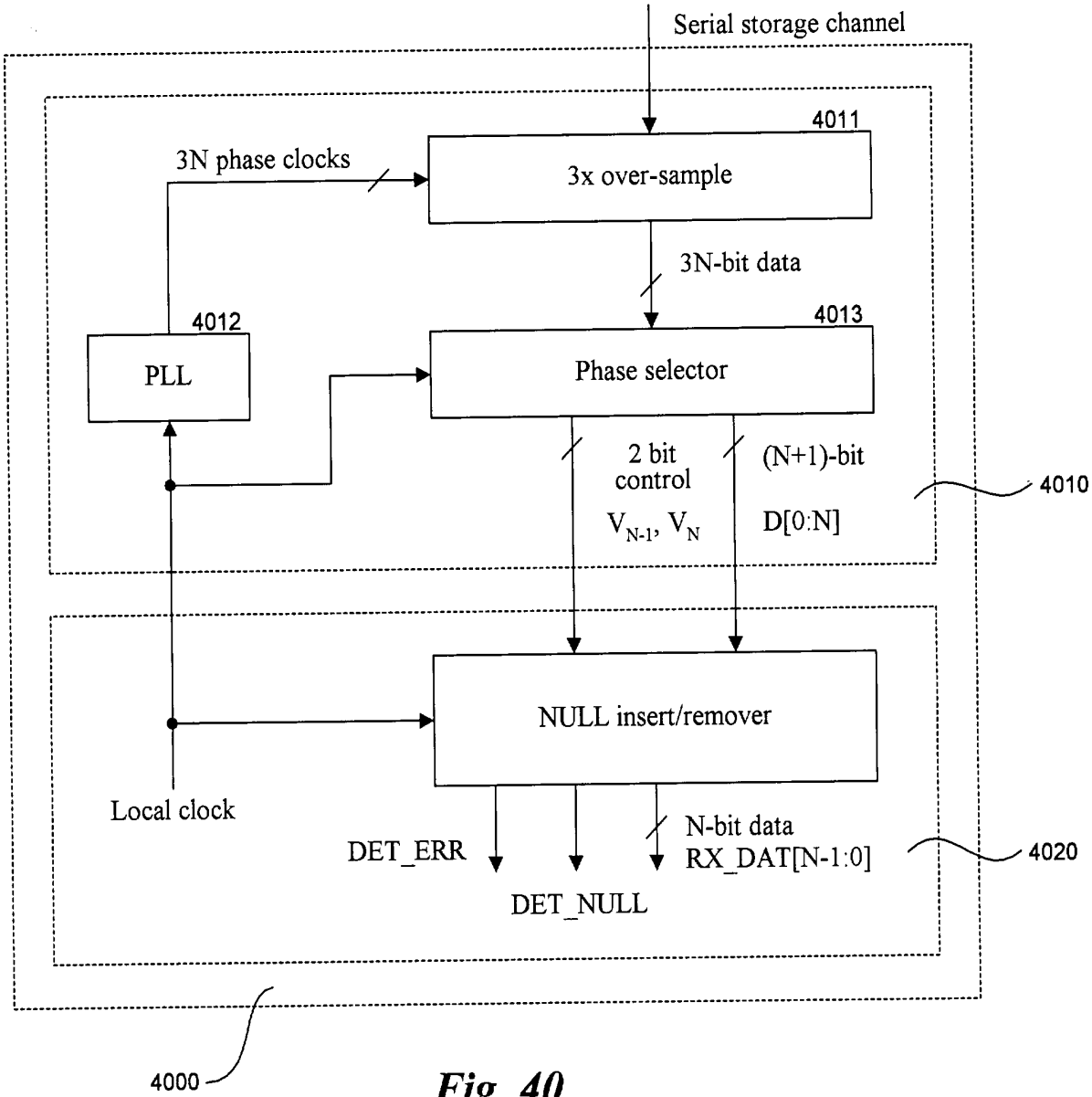

Serial storage channel
4011
3N phase clocks
3x over-sample
3N-bit data
4012
4013
PLL
Phase selector
2 bit control
V_{N-1}, V_N
(N+1)-bit
D[0:N]
4010
NULL insert/remover
Local clock
DET_ERR
DET_NULL
N-bit data
RX_DAT[N-1:0]
4020
4000

Fig. 40

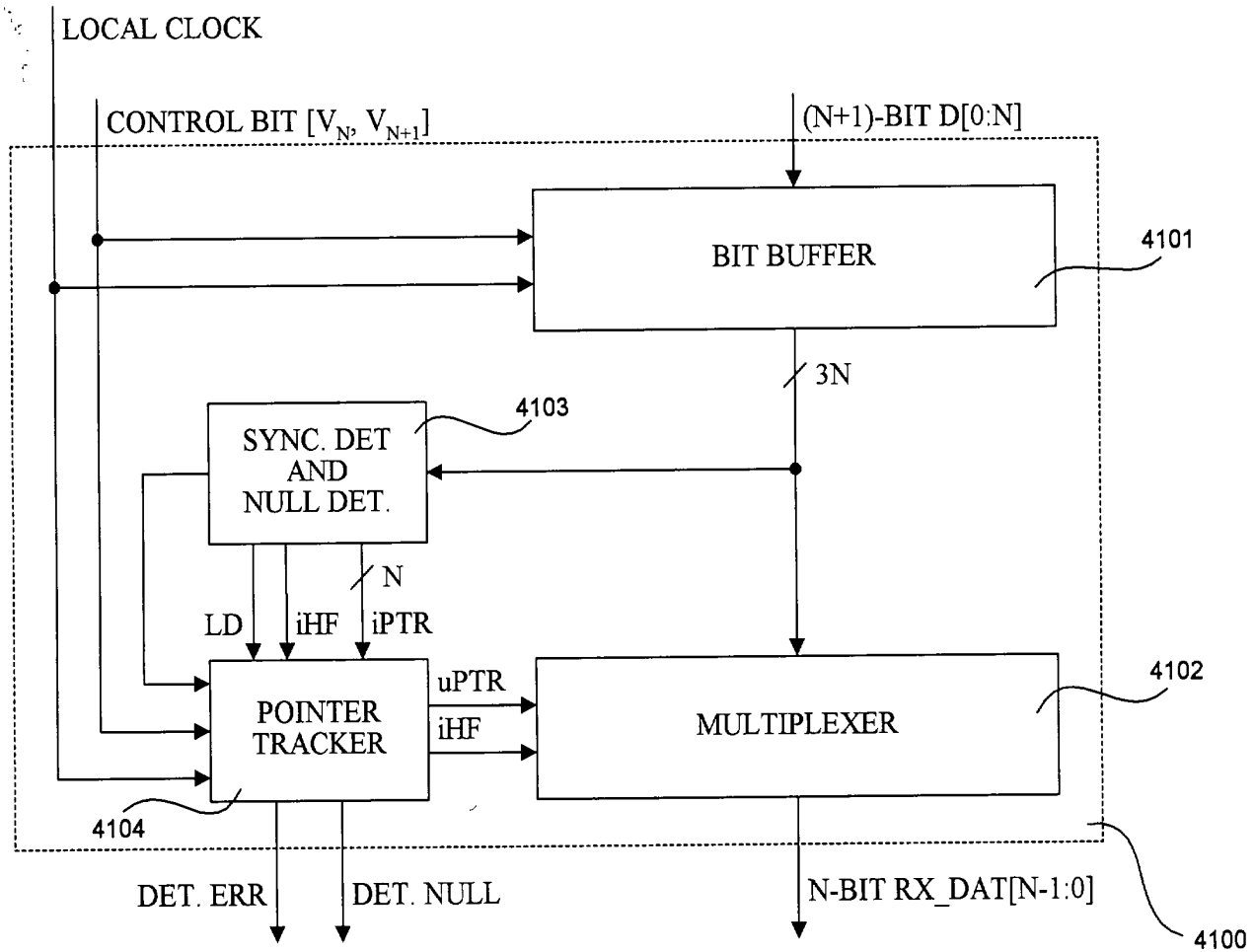

LOCAL CLOCK
CONTROL BIT [V_N, V_{N+1}]
(N+1)-BIT D[0:N]
BIT BUFFER
4101
3N
SYNC. DET AND NULL DET.
4103
N
LD
iHF
iPTR
POINTER TRACKER
uPTR
iHF
MULTIPLEXER
4102
4104
DET. ERR
DET. NULL
N-BIT RX_DAT[N-1:0]
4100

Fig. 41

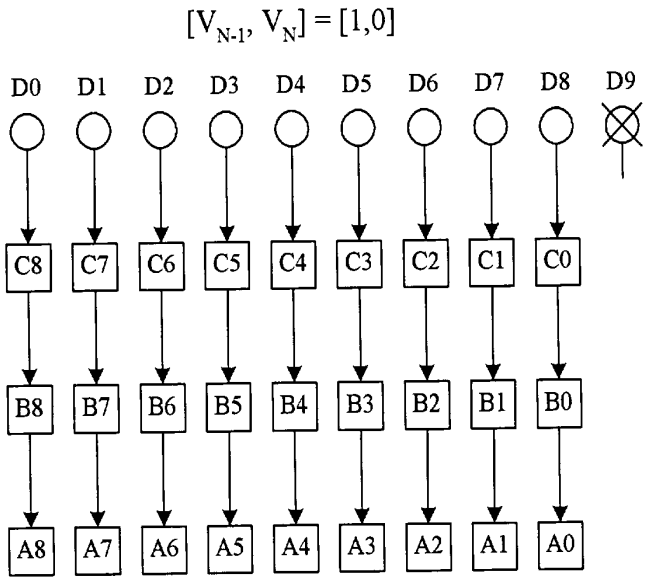

$[V_{N-1}, V_N] = [1,0]$
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
C8
C7
C6
C5
C4
C3
C2
C1
C0
B8
B7
B6
B5
B4
B3
B2
B1
B0
A8
A7
A6
A5
A4
A3
A2
A1
A0

Fig. 42A

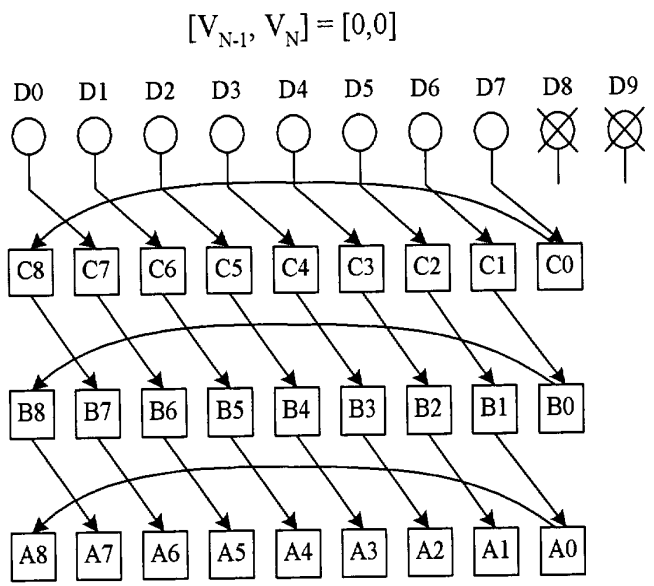

$[V_{N-1}, V_N] = [0,0]$
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
C8 C7 C6 C5 C4 C3 C2 C1 C0
B8 B7 B6 B5 B4 B3 B2 B1 B0
A8 A7 A6 A5 A4 A3 A2 A1 A0

Fig. 42B

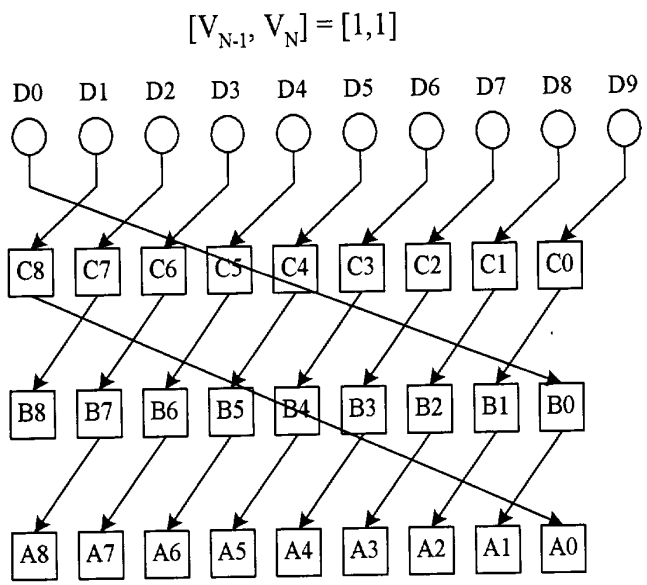

$[V_{N-1}, V_N] = [1,1]$
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
C8 C7 C6 C5 C4 C3 C2 C1 C0
B8 B7 B6 B5 B4 B3 B2 B1 B0
A8 A7 A6 A5 A4 A3 A2 A1 A0

Fig. 42C

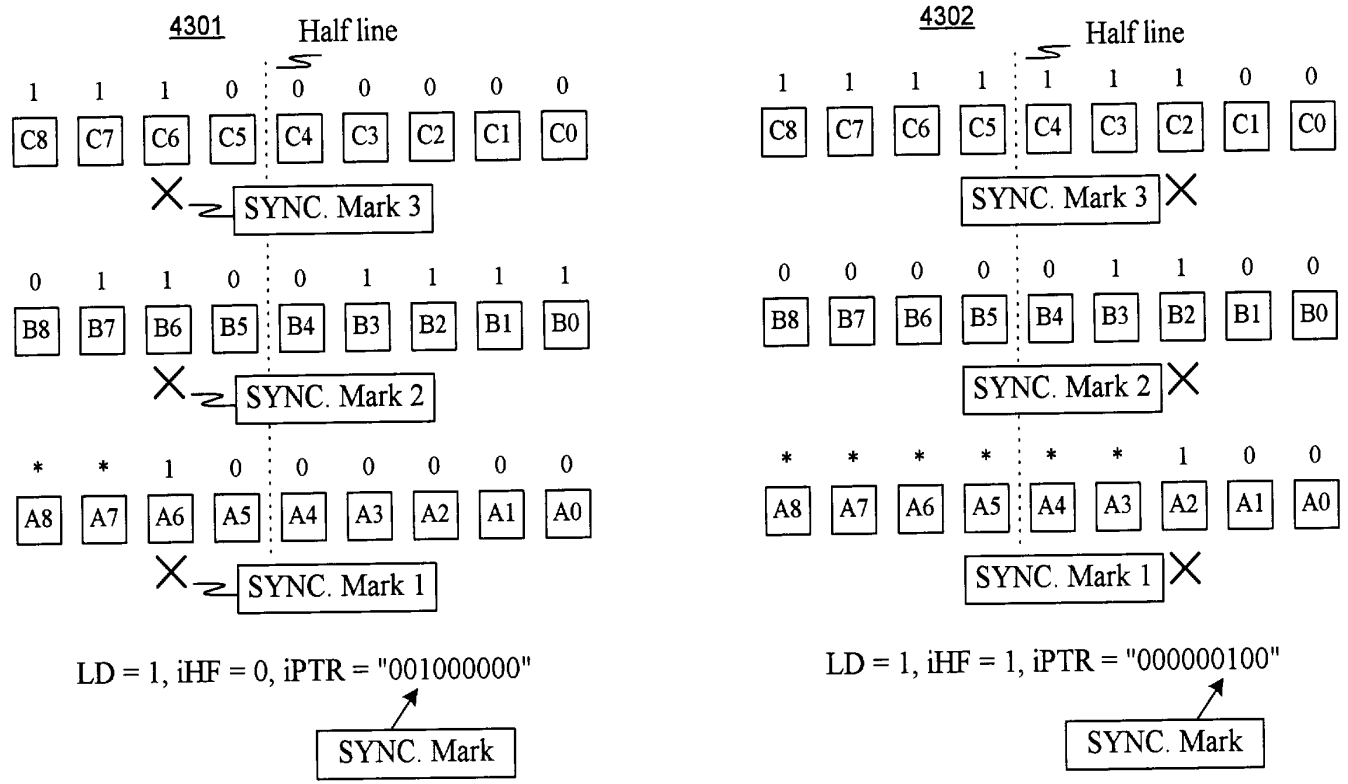
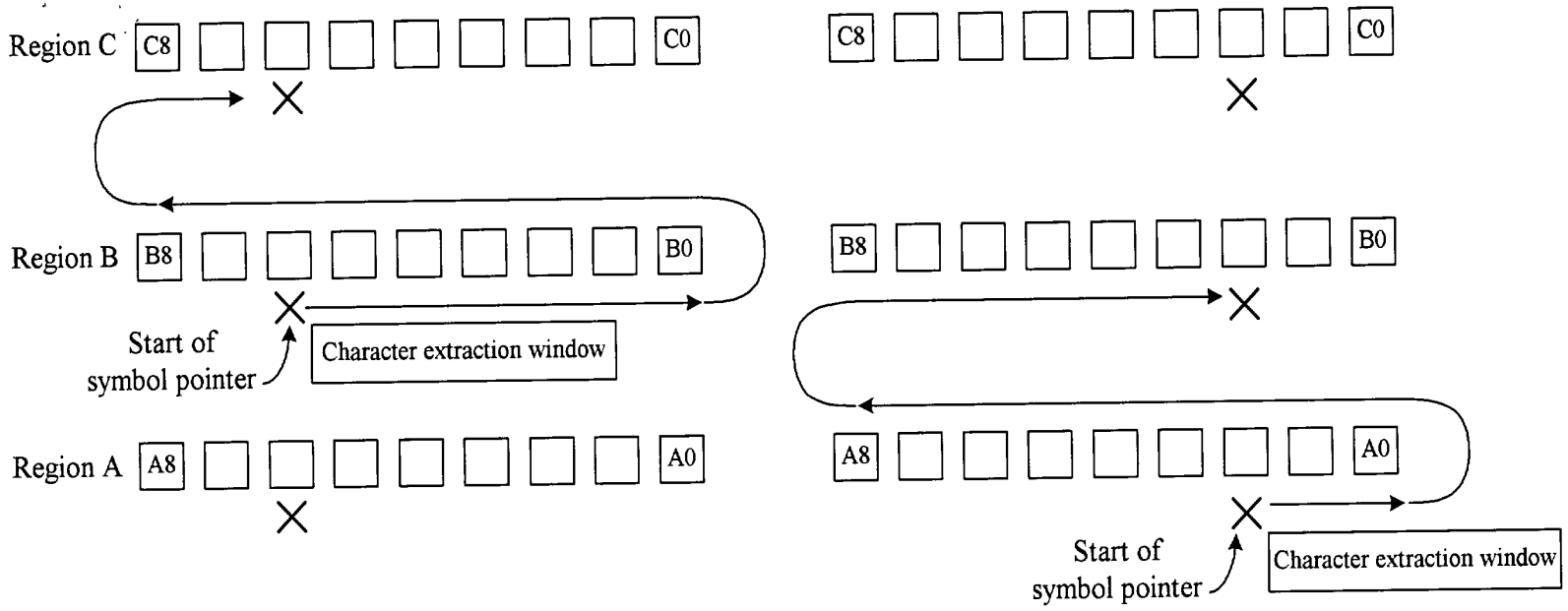

4301
Half line
1 1 1 0 0 0 0 0 0
C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC. Mark 3
0 1 1 0 0 1 1 1 1
B8 B7 B6 B5 B4 B3 B2 B1 B0
SYNC. Mark 2
* * 1 0 0 0 0 0 0
A8 A7 A6 A5 A4 A3 A2 A1 A0
SYNC. Mark 1
LD = 1, iHF = 0, iPTR = "001000000"
SYNC. Mark
4302
Half line
1 1 1 1 1 1 1 0 0
C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC. Mark 3
0 0 0 0 0 1 1 0 0
B8 B7 B6 B5 B4 B3 B2 B1 B0
SYNC. Mark 2
* * * * * * 1 0 0
A8 A7 A6 A5 A4 A3 A2 A1 A0
SYNC. Mark 1
LD = 1, iHF = 1, iPTR = "000000100"
SYNC. Mark

Fig. 43



LD = 1, iHF = 0, iPTR = "001000000"

LD = 1, iHF = 1, iPTR = "000000100"

Fig. 44

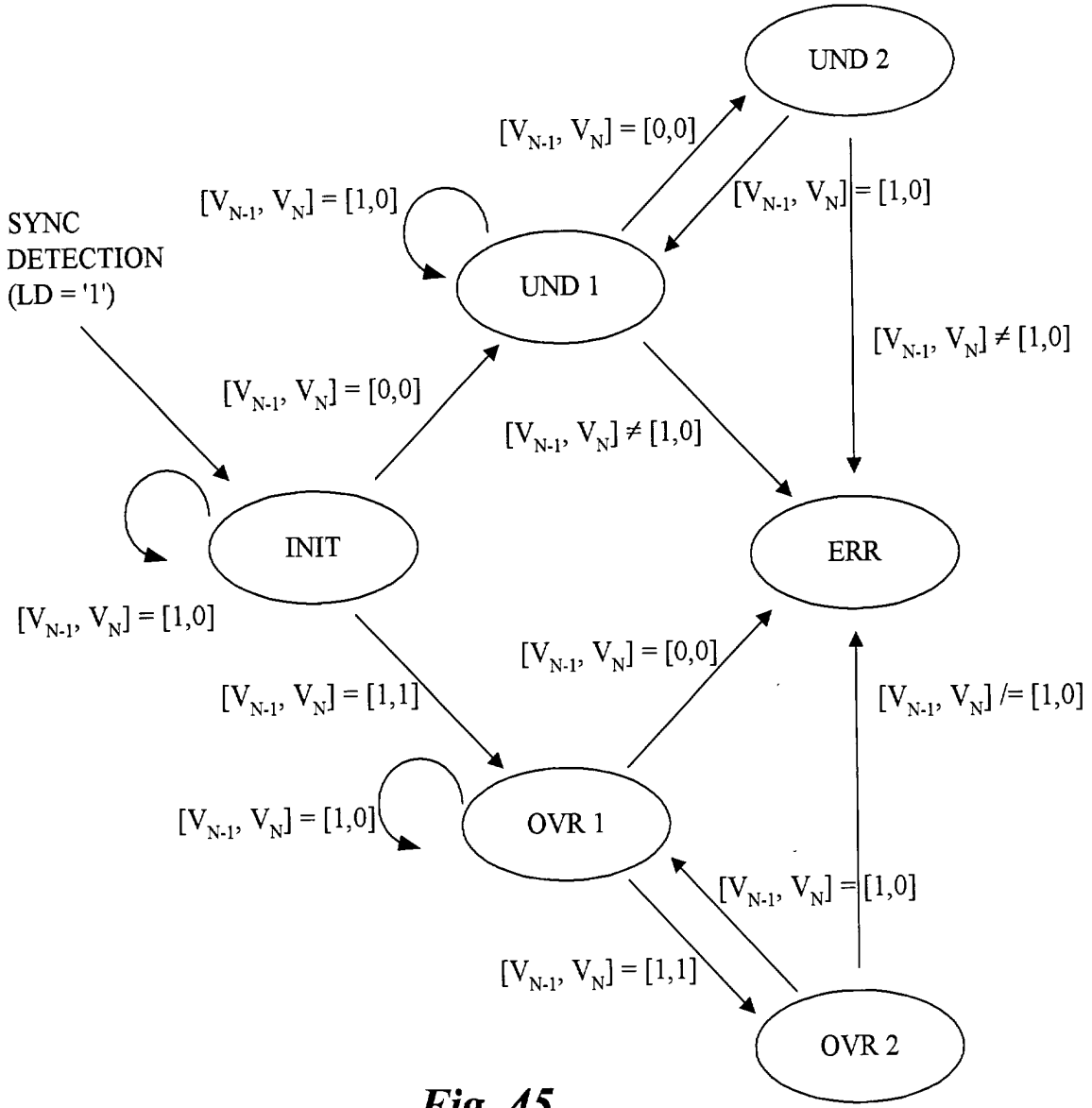

UND 2
$[V_{N-1}, V_N] = [0,0]$
$[V_{N-1}, V_N] = [1,0]$
$[V_{N-1}, V_N] = [1,0]$
SYNC
DETECTION
(LD = '1')
UND 1
$[V_{N-1}, V_N] \neq [1,0]$
$[V_{N-1}, V_N] = [0,0]$
$[V_{N-1}, V_N] \neq [1,0]$
INIT
ERR
$[V_{N-1}, V_N] = [1,0]$
$[V_{N-1}, V_N] = [0,0]$
$[V_{N-1}, V_N] = [1,1]$
$[V_{N-1}, V_N] /= [1,0]$
$[V_{N-1}, V_N] = [1,0]$
OVR 1
$[V_{N-1}, V_N] = [1,0]$
$[V_{N-1}, V_N] = [1,1]$
OVR 2

Fig. 45

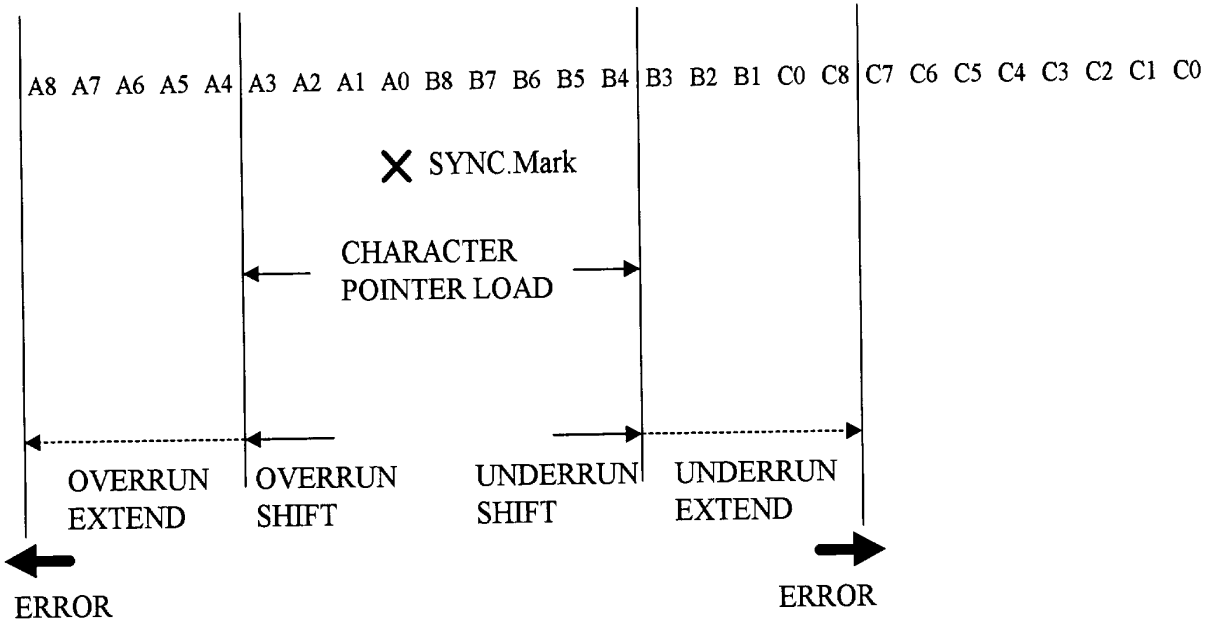

A8 A7 A6 A5 A4 A3 A2 A1 A0 B8 B7 B6 B5 B4 B3 B2 B1 C0 C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC.Mark
CHARACTER
POINTER LOAD
OVERRUN
EXTEND
OVERRUN
SHIFT
UNDERRUN
SHIFT
UNDERRUN
EXTEND
ERROR
ERROR

Fig. 46

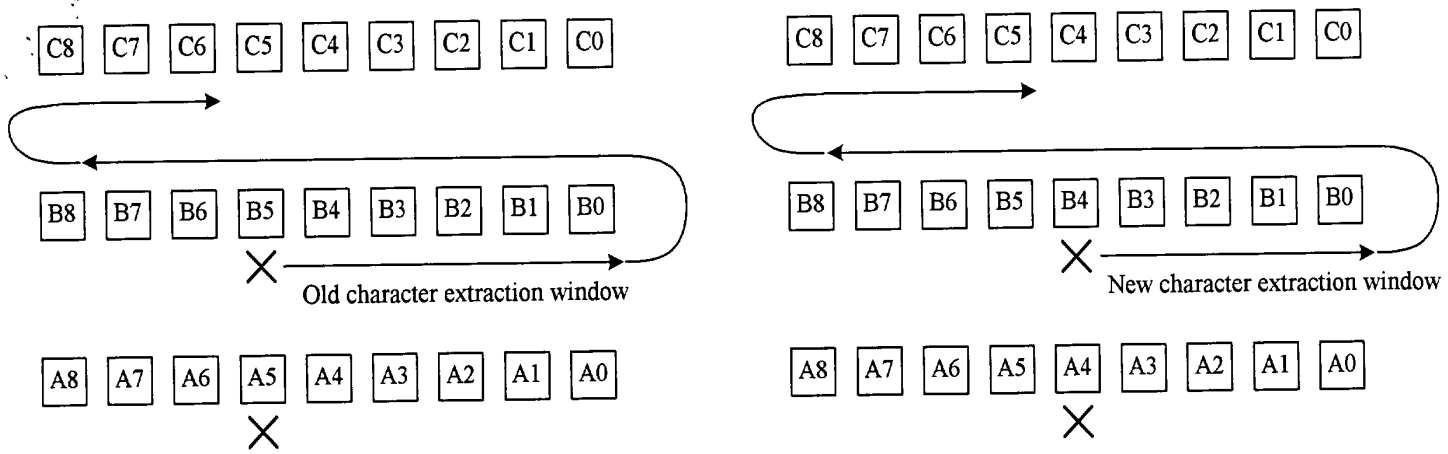


Fig. 47A

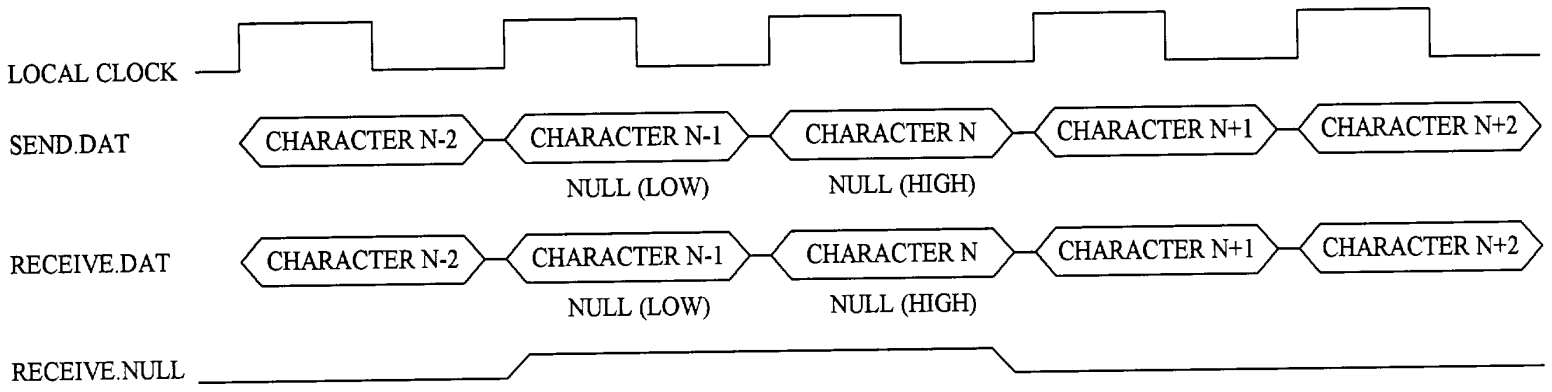


Fig. 47B

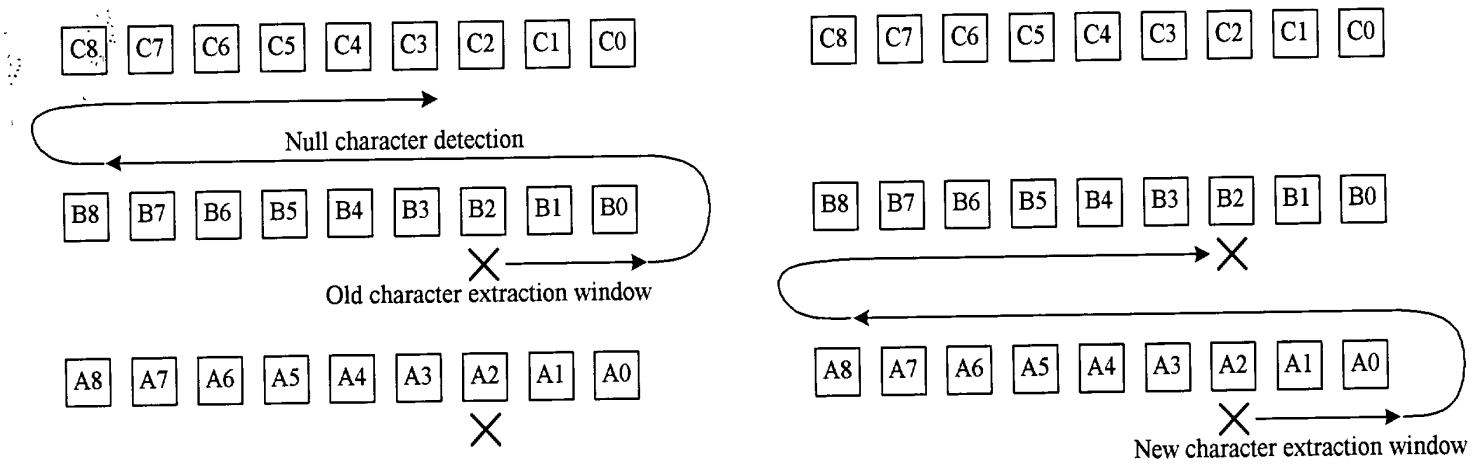


Fig. 48A

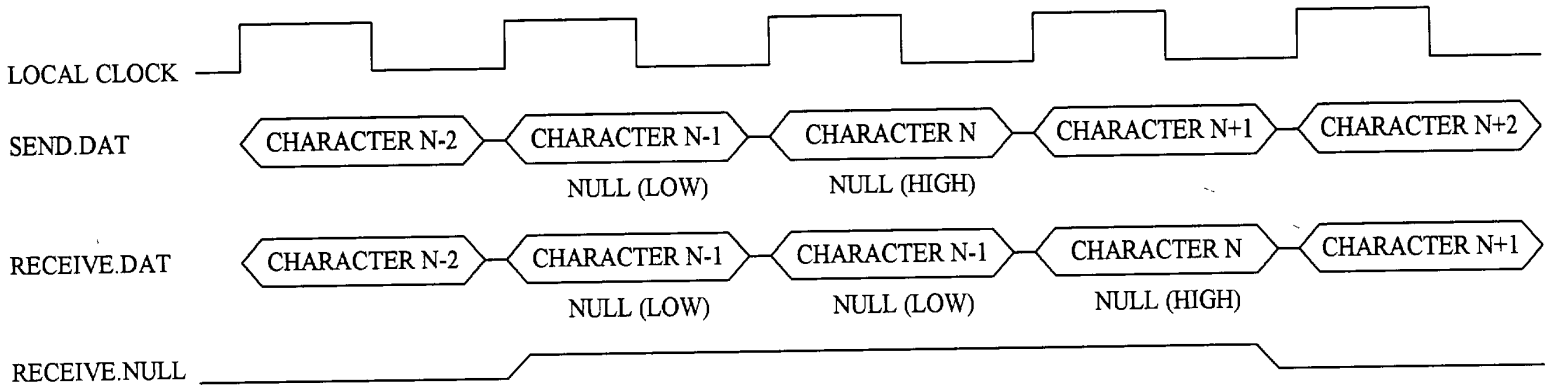


Fig. 48B

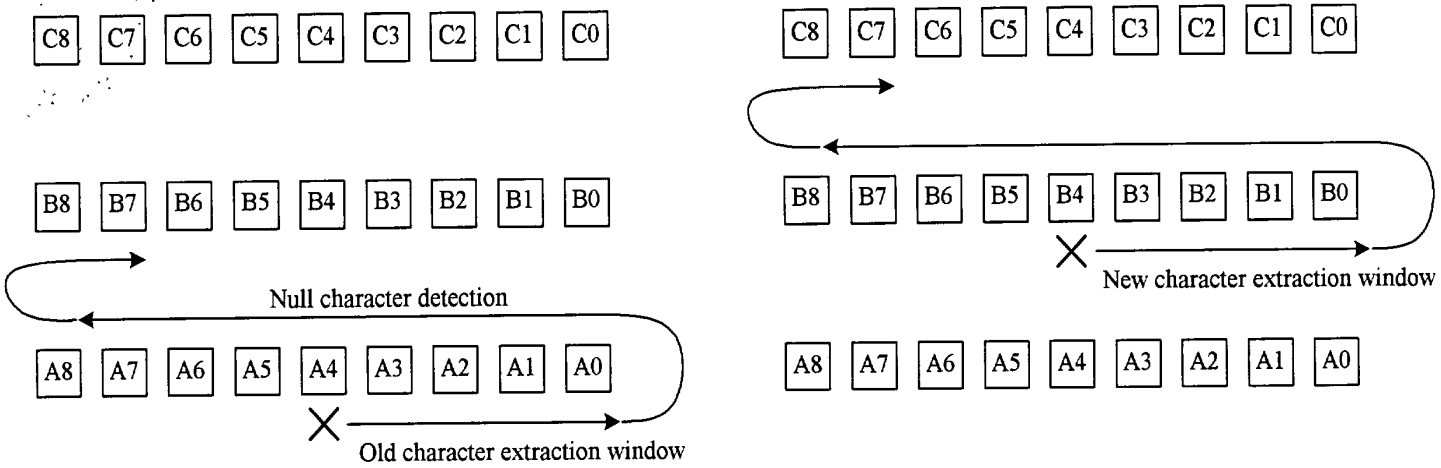


Fig. 49A

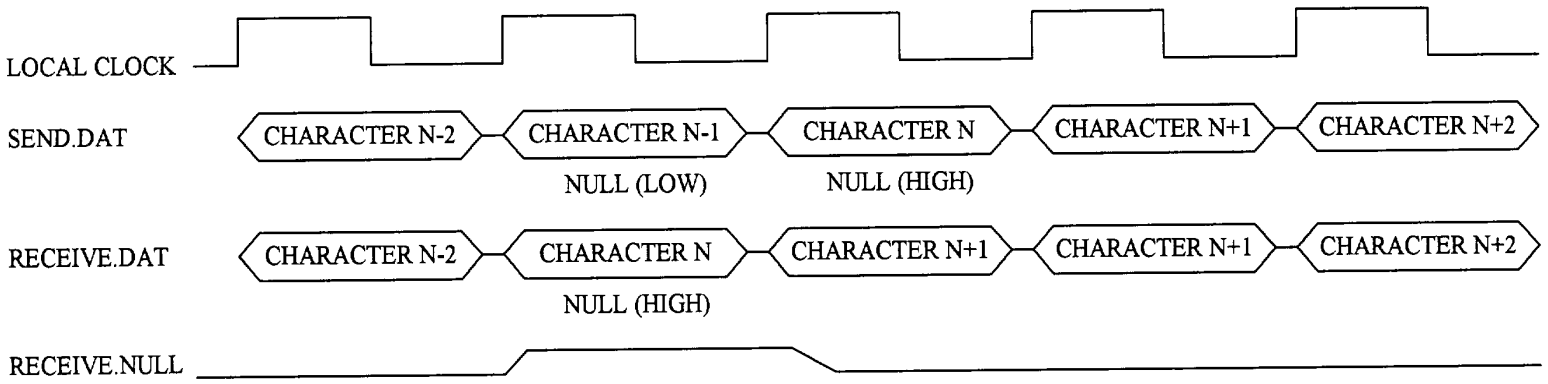


Fig. 49B